

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-081401

(43)Date of publication of application : 28.03.1997

(51)Int.Cl. G06F 9/46
G06F 9/46

(21)Application number : 07-239516

(71)Applicant : HITACHI LTD

(22)Date of filing : 19.09.1995

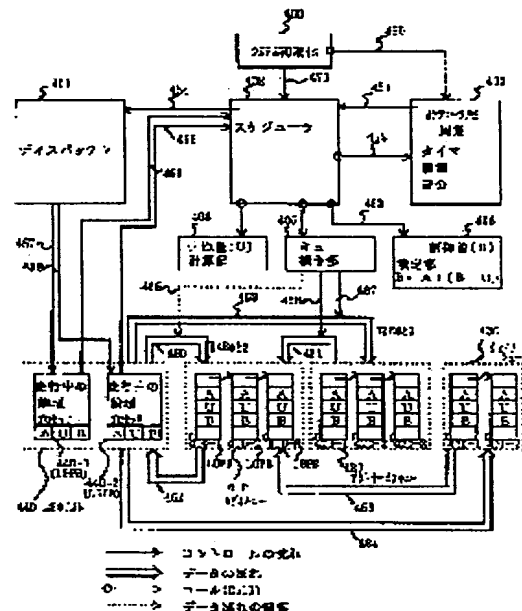
(72)Inventor : UMENO HIDENORI
AMANO HIDEAKI
YAMAMOTO YASUO

(54) GLOBAL RESOURCE CAPPING METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To designate which of a global or local resource capping method is applied or not by limiting the CPU utilization factor of each virtual computer to a limit value or below in each small section.

SOLUTION: A period timer interruption generation part 403 divides actual time into sections (about a second), and every time the section passes, a timer interruption is generated. A scheduler 402 updates a control value B by an expression $B = \text{initial control value } A + (B - U)$. An interruption processing is performed for a running logical processing by the scheduler 402, service amount U is updated, further, a queuing is performed for a ready-queue 410 or an out service queue 420 by the comparison of the service amount U after the update and the control value B. Subsequently, the processing of the setting part 406 of the control value B is performed. A control value B setting part 406 controls CPU service amount so as to be an initial control value A or below by fluctuating the control value B to be the upper limit value of the service amount of each logical processor for every section.



BEST AVAILABLE COPY

LEGAL STATUS

[Date of request for examination] 13.01.1999

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3543441

[Date of registration] 16.04.2004

[Number of appeal against examiner's decision of rejection]

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平9-81401

(43)公開日 平成9年(1997)3月28日

(51)Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/46	3 5 0		G 0 6 F 9/46	3 5 0
	3 4 0			3 4 0 D

審査請求 未請求 請求項の数2 O L (全 24 頁)

(21)出願番号 特願平7-239516

(22)出願日 平成7年(1995)9月19日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 梅野 英典

神奈川県秦野市掘山下1番地株式会社日立
製作所汎用コンピュータ事業部内

(72)発明者 天野 英昭

神奈川県秦野市掘山下1番地株式会社日立
製作所汎用コンピュータ事業部内

(72)発明者 山本 保男

東京都千代田区内神田二丁目14番6号日立
電子サービス 株式会社

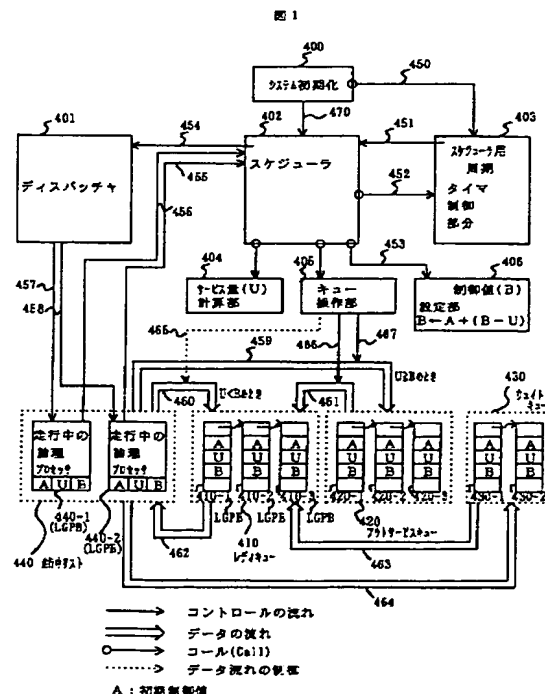
(74)代理人 弁理士 小川 勝男

(54)【発明の名称】 大域的なりソースキャッピング方法

(57)【要約】

【目的】VMのCPU利用率を、ユーザの指定量に抑える方式であるリソースキャッピングにおいて、短期(秒オーダ)の実時間ではなく、長期の実時間(分または時間のオーダ)において、指定量に抑える方式を提供する。

【構成】実時間において、小区間(例えば1秒)ごとにハイパバイザに割り込みを入れる手段、各小区間ごとに各VMのCPU使用時間を測定する手段、各小区間において各VMのCPU利用率を制限値以下に制限する手段からなる。



【 特許請求の範囲】

【 請求項1 】 一台の実計算機システムの下で複数台のオペレーティングシステム(OS) を動かすことのできる仮想計算機システムにおいて、各OS のCPU利用率をある一定の値以下に制限するリソースキャッピング方法であって、各OS のCPUサービス量を指定するステップと、それに基づいて各OS のCPUサービス量の上限値である初期制御値を計算するステップと、実時間である観測時間を小さな時間である小区間に分割するステップと、ある上記のOS のCPU利用率を、各該小区間において、ある制御値以下に抑える場合、該制御値を該OS の直前の小区間におけるCPU利用率によって動的に変動させるステップとにより、ある小区間においては、該OS のCPU利用率が初期制御値を越えることを許すが、該OS の全観測時間におけるCPU利用率を初期制御値以下に抑える大域的なりソースキャッピング方法。

【 請求項2 】 一台の実計算機システムの下で複数台のオペレーティングシステム(OS) を動かすことのできる仮想計算機システムにおいて、各OS のCPU利用率をある一定の値以下に制限するリソースキャッピング方法であって、各OS のCPUサービス量を指定するステップと、それに基づいて各OS のCPUサービス量の上限値である初期制御値を計算するステップと、実時間である観測時間を小さな時間である小区間に分割するステップと、ある上記のOS に対しては、そのCPU利用率を、各該小区間において、常に初期制御値以下に抑える局所的なりソースキャッピング方法を適用し、別のOS に対しては、そのCPU利用率をある制御値以下に抑える場合、該制御値を該OS の直前の小区間におけるCPU利用率によって動的に変動させるステップとにより、ある小区間においては、該OS のCPU利用率が初期制御値を越えることを許すが、該OS の全観測時間におけるCPU利用率を初期制御値以下に抑える大域的なりソースキャッピング方法。

【 発明の詳細な説明】

【 0 0 0 1 】

【 産業上の利用分野】 本発明は、一台の実計算機システムの下で複数台の仮想計算機(VM: Virtual Machine) を同時に動かすことのできる仮想計算機システムのCPUスケジュール方式に関する。各仮想計算機は、該実計算機と同等の機能を持ち、実計算機上で動くのと同じOS を動作させることができる。この意味で、仮想計算機システムは一台の実計算機システムの下で複数台のOS を同時に動かすことのできるシステムであるとも云える。仮想計算機システムを動かす実計算機システムをホスト実計算機システムと呼ぶ。CPUスケジュール方式とは、同時に動作している各VMに、いつ、どの程度のCPUサービス量を与えるかということを決定する方法である。

【 0 0 0 2 】

【 従来の技術】 仮想計算機システムの普及に伴い、その性能を向上させるために、いろいろなハードウェアによる支援機構が適用されてきた。仮想計算機システムは、ハイパバイザと呼ばれる制御プログラムを含む。ハイパバイザは、システムの主記憶、CPU、I/O系というシステムリソースを各VMに与え、各VMを同時に動かすためにCPUスケジュールを行う。すなわち、ハイパバイザは、そのCPUスケジュールにおいて、各VMの状態を管理し、それをいつ動かすか、それにどのくらいのCPUサービス量を与えるかを決定する。

【 0 0 0 3 】 仮想計算機システムを動かす実計算機システムは、マルチプロセッサシステムでも、ユニプロセッサシステムでも良い。ここでいう、マルチプロセッサシステムとは、主記憶装置を共有する2 台以上のプロセッサからなり、ユニプロセッサシステムは1 台のプロセッサからなる。ここでいうプロセッサとは主記憶に格納されたプログラムを構成する機械命令をフェッチして実行していく機能を有する処理装置のことである。

【 0 0 0 4 】 仮想計算機VMの構成を図3 に示すが、詳しくは後述する。仮想計算機VMは、それ自体ユニプロセッサモードまたは、マルチプロセッサモードであり得る。実計算機システムとの対応で云うと、ホスト実計算機システムがVMに対応し、実プロセッサに対応するものを論理プロセッサと呼ぶ。すなわち、ユニプロセッサモードのVMは論理プロセッサ1 台からなり、マルチプロセッサモードのVMは、2 台以上の論理プロセッサからなり、各論理プロセッサはVMにとっての主記憶領域を共有する。VMの主記憶領域は、ホスト実計算機システムの主記憶領域のある部分連続領域を、その主記憶領域として割り当てる場合もあるし、ハイパバイザが構成する仮想空間を、その主記憶領域として使用する場合もある。

【 0 0 0 5 】 従来技術において、各VMにサービス率を指定する機能がある。この機能は、以下のとおりである。たとえば、VM1、VM2、...VMN というN台(N は1 以上) の仮想計算機が同時に動作しているとしよう。その各々にサービス率S 1、S 2、...、S N を指定したとする。各サービス率S i は、0以上である。このサービス率S 1、S 2、...、S N は、各仮想計算機VM1、VM2、...VMNに割り当てるCPUサービス量を表す。さらに正確に云えば、各VMは、待ち状態になることなく、CPUを与えられれば、いつでも、それを使い続けることができる状態(これをReady状態という) にあるとした場合、このサービス率指定は、各VM1、VM2、...VMNに与えられるCPUサービス量の比がS 1、S 2、...、S Nとなるようにスケジュールすることを要求するものである。

【 0 0 0 6 】 さらに従来技術においては、リソースキャッピング(resource capping) という機能がある。これについて以下に説明する。 サービス率制御において

3

は、各VMが常にReady状態または走行状態であるとき、そのサービス量の比が、指定サービス率の比になるように制御するものである。しかし、実際は、各VMは、待ち状態になり得る。例えば、VM上のOSがI/O動作を要求し、その完了を待って待ち状態になる場合が、これに該当する。たとえば、今、VM1が、サービス率S1により規定されるサービス量A1に達すること無く、待ち状態になったとしよう。そうすると、本来VM1に与えられるべきCPUサービスが、特に他の指定も無い場合は、他のVM2、・・・VMNに割当らることになる。そうすると、そのVM2、・・・VMNの中でCPUを大量に使う負荷を持つもの(例えば、VM2とする)は、指定サービス率S2によって規定されるCPUサービス量A2を越えてサービスされることになる。

【0007】たとえば、リソースキャッピングの例を、2台の仮想計算機VM1、VM2について考える。各サービス量をS1=60、S2=40とする。ホスト実計算機システムでの実プロセッサ台数を2台とし、その全処理能力を200としたとき、VM1に与えられる規定CPUサービス量A1は、CPU利用率にして2台の実プロセッサ合計で120%(すなわち1台の実プロセッサ当たり平均60%)、VM2に与えられる規定CPUサービス量A2は、CPU利用率にして2台の実プロセッサ合計で80%(すなわち1台の実プロセッサ当たり平均40%)、ということになる。しかし、もし、VM2が待ち時間が多くて、CPU利用率が、2台の実CPU全体で50%にしか達せず、VM1が常にReady状態、または、走行状態の負荷を持つとすると、本来VM2に与えられるべき30%

のCPU利用率が、VM1に与えられ、そのCPU利用率が150%に達することとなる。これは、システム全体の性能向上としては、当然の処理方式であるが、VM1のユーザとしてはCPU利用率120%分の費用しか支払わないという契約を結んで使用したいというユーザもいる。このようなユーザに対しては、システムとして、VM1のCPU利用率を120%以下に制限する必要がある。この制限機能が、リソースキャッピングと呼ばれるものである。

【0008】この従来のリソースキャッピングの問題点について以下に述べる。あるVMのCPU利用率を規定値A%(Aは0より大きな値とする)以下に抑えるということは、正確に考えると、二つの意味、すなわち、局所的に抑えると云う意味と、大域的に抑えると云う意味がある。この違いを図39、図40、図41を用いて説明する。図39は、あるVMのある負荷の本来のCPU利用率を実時間の経過を横軸にとって表したものである。90がそのCPU利用率の推移を表す。値A(91)(%)が目的とする制御値であるとする。この負荷は時刻Toを境として、Toまでは制御値Aより、そのCPU利用率が低く、Toから移行は制御値Aより高いとする。局所的に抑えるということは、観測時間(T1)

4

(例えば1時間)を各小区間(例えば1秒)に分割した場合、全ての小区間でのCPU利用率をA%以下に抑えるということである。したがって、この局所的リソースキャッピングを適用すると、この負荷のCPU利用率の経過は図40に示すようになる。即ち時刻To移行は制御値Aで抑えられることになる。時刻Toより前では、元々、制御値Aより低いので、そのまま低いままである。これを全体の観測時間T1で見るとこの負荷の全体のCPU利用率は、制御値Aよりかなり低い値になってしまう。一方、大域的に抑えるということは、局所的にA%を超えることがあっても良いが、長期の観測時間全体においてCPU利用率をA%以下に抑えるということである。この負荷に大域的リソースキャッピングを適用した場合のCPU利用率の経過を図41に示す。ここでは、Toまでが制御値Aより低いので、その不足分をTo以降の制御値に加算することにより、To以降では、制御値Aを超えることを許しているが、全体の観測時間T1で見れば、全体のCPU利用率はAで抑えられ、しかも、かなりAに近い値にまで達することができる。このように局所的な意味で指定するか、大域的な意味で指定するかは、ユーザによるところである。しかし、従来のリソースキャッピングは、局所的な意味でしか指定できなかった。この局所的なリソースキャッピングについては、以下のような問題点がある。

【0009】たとえば、CPU利用率を50%に抑えたい(A=50)とする。ある負荷を処理したとき、観測区間Tを、二つの同一長さの区間T1、T2に分割したとする。このとき、区間T1で、この負荷の本来の(リソースキャッピング制御を行わないときの)CPU利用率が30%あるとすると、区間T1では、局所リソースキャッピングの適用、不適用にかかわらずCPU利用率は30%となる。次の区間T2で、たとえば該負荷のCPU利用率が本来90%有るものとする、局所リソースキャッピングを適用すれば、CPU利用率は50%に制限される。したがって、観測区間Tを通して考えると、CPU利用率は40%に抑えられることになる。これは、ユーザから見れば抑え過ぎであるという意見もあるであろう。なぜなら、そのようなユーザにとっては、観測区間Tを通しての本来のCPU利用率は60%であり、それを、大域的に50%に制限するだけで良いという意見だからである。このようなユーザに対しては、大域的なリソースキャッピングを行う必要があるが、従来方式では、その手段がない。

【0010】

【発明が解決しようとする課題】本発明の目的は、あるVMのCPU利用率を、ユーザの指定量に抑える方式であるリソースキャッピングにおいて、短期(秒のオーダー)の実時間ではなく、長期の実時間(分または時間のオーダー)において、該VMのCPU利用率を該指定量に抑えるという方式を提供することである。各短期の区間

5

においては、該指定量を越えることが有っても良い。この方式を大域的なりソースキャッピングと呼ぶ。

【 0 0 1 1 】これに対し、従来のリソースキャッピングは、全ての短期区間において、該CPU利用率を指定量以下に抑えるという局所的なりソースキャッピングである。

【 0 0 1 2 】本発明の別の目的は、各VM毎に、上記の大域的なりソースキャッピング方式を適用するか、局所的なりソースキャッピング方式を適用するかを指定可能とすることである。

【 0 0 1 3 】

【課題を解決するための手段】実時間において、小区間（例えば1秒）ごとに、ハイパバイザに割り込みを入れる手段、各小区間毎に、各VMのCPU使用時間を測定する手段、各小区間において各VMのCPU利用率の制限値を動的に指定する手段、各小区間において各VMのCPU利用率を該制限値以下に制限する手段からなる。

【 0 0 1 4 】さらに、ユーザから、システム全体、または、各VMごとにCPUサービス率を指定する手段、システム全体、または、各VM毎に、局所的なりソースキャッピングを適用するの、大域的なりソースキャッピングを適用するのかを指示する手段からなる。

【 0 0 1 5 】

【実施例】図1 は、本発明の基本的な構成図である。400 はシステム初期化を行う処理プログラムである。これは、ハイパバイザを立ち上げるときに、システムで使用するデータを初期化する（詳細は後述する）。その初期化の一環として、周期タイマ割込み発生部403をコール（call）する。403 は、周期タイマと、それを制御するプログラムからなる。システム初期化400 は、線450 が示す様に、周期タイマ割込み発生部403をコールし、全命令プロセッサに対して、周期的に、また、同期してタイマ割込みを発生するように設定する。そのタイマ割込みの周期 Δt は約1秒である。この周期はユーザが設定することもできる。この周期を区間と呼ぶ。400 はシステム初期化が終わった後、線470 が示すように、スケジューラ402へ制御を渡す。スケジューラ402 は、ハイパバイザの主要なプログラムである。スケジューラ402 は、各仮想計算機VMの各論理プロセッサをスケジュールする。すなわち、走行可能となった論理プロセッサをレディキュー（Ready Queue）410にキューイングすると共に、レディキュー410に登録された走行可能な論理プロセッサを選択して、線454 が示すように、ディスパッチャ401に制御を渡す。ディスパッチャ401 は、その選ばれた論理プロセッサ440-1または、440-2に、線457 または、線458 が示すように制御を渡し、走行させる。仮想計算機VMは、図3 に示す構成を持つが詳しくは後述する。仮想計算機VMは、いくつかの（1台でも良い）論理プロセッサからなる。ハイパバイザは、論理

6

プロセッサの状態を管理するために、論理プロセッサタスク制御ブロックLGPB(Logical processor task block)を構成する（図7 に詳細を示し、後述する）。440 は、走行中の論理プロセッサのリストであるが、440-1、440-2 は、このLGPBであるが、走行中の論理プロセッサも表すものである。論理プロセッサが、走行可能ではあるが命令プロセッサがビジー（busy）であるが故に、そのサービスを待っている状態であるとき、これをレディー状態という。レディー状態の論理プロセッサに対応する制御ブロックLGPBが、レディキュー410にキューイングされる。410-1、410-2、410-3 は、この制御ブロックLGPBを表す。走行中の論理プロセッサが、或事象の発生を待つことが必要になったとき、その論理プロセッサは、ウェイト（wait）状態、すなわち、待ち状態に入る。待ち状態の論理プロセッサのLGPBは、ウェイトキュー（Wait Queue）430にキューイングされる。430-1、430-2 がこの制御ブロックLGPBである。各論理プロセッサタスク制御ブロックLGPBには、論理プロセッサのリソースデータ（レジスタや、PSWなど）の他に、命令プロセッサのサービスを受けた割合を表す値（U：これをCPUサービス量と呼ぶ）と、リソースキャッピングの指定により、各論理プロセッサに対して決定された該サービス量の上限値AとBが含まれる。これらを制御値A、Bと呼ぶ。

【 0 0 1 6 】制御値Aは、初期制御値であり、ユーザの指定値、アクティブな論理プロセッサの台数により、VMのアクティブ時に計算され、その後、新たに、VMがアクティブまたはディアクティブされない限り一定である。一方制御値Bは、動的に変動する値である。406 は、制御値B設定部であり、周期的に、この制御値Bを計算し直すプログラムである。

【 0 0 1 7 】この406の制御方法が、従来の方法と異なり、新方式を与え、本発明の主要部分を構成するものである。すなわち、各論理プロセッサのサービス量Uは、走行中の論理プロセッサが何らかの理由で中断されて、線455または線456 が示すようにコントロールがスケジューラ402に移ったとき、スケジューラ402は、該中断された論理プロセッサのCPUサービス量Uを計算し直す。このためにスケジューラ402は、サービス量計算部404をコールする。404は、中断されるまでに走行した命令プロセッサによるサービス時間を加算して、現在までのCPUサービス量Uを該論理プロセッサについて求め、その制御ブロックLGPBに設定し直す。さらに、ある論理プロセッサが中断されて、コントロールがスケジューラ402に移ったとき、そのサービス量Uを上記のように更新した後、スケジューラ402は、キュー操作部405をコールする。キュー操作部405は、該制御ブロックLGPB内の現在の該CPUサービス量Uを該制御値Bと比較する。キュー操作部405は、このとき $U < B$ であれば、いまだ制限値Bに達し

7

ていないので、該論理プロセッサタスク制御ブロック LGPBをレディキューにキューイングする。この制御ブロック LGPBの動きを表したものが線4 6 0 であり、この動きを制御する働きが線4 6 5 である。キュー操作部4 0 5 は、そうでないとき、すなわちサービス量Uが制御値B以上となったとき、該論理プロセッサのサービス量は制御値Bに達したので、該論理プロセッサをアウトサービス状態とする。さらに、アウトサービス状態の論理プロセッサの制御ブロック LGPBは、ディスパッチの対象からはずすために、アウトサービスキュー4 2 0 にキューイングする。このデータの動きが線4 5 9 で表現される。さらに、このキューイングを制御する働きを線4 6 7 で表す。制御ブロック LGPBである4 2 0 - 1、4 2 0 - 2、4 2 0 - 3 は、このアウトサービスキュー4 2 0 内の制御ブロック LGPBである。4 0 3 は、周期タイマ割り込み発生部であり、実時間を区間(一秒程度)に分割して、その区間が経過する度にタイマ割り込みを発生させる。これを、スケジューラタイマ割り込みという。このスケジューラタイマ割り込みの発生により、コントロールが線4 5 1 が示すようにスケジューラ4 0 2 に移る。スケジューラ4 0 2 は、制御値Bを以下の式により更新する。

【0018】 $B = A + (B - U)$

この更新はレディキュー4 1 0 の全ての論理プロセッサタスク制御ブロック LGPBについて実行される。さらに、この更新は、アウトサービスキュー4 2 0 の全ての論理プロセッサタスク制御ブロック LGPBについて実行される。さらに、この更新はウェイト・キュー4 3 0 の全ての論理プロセッサタスク制御ブロック LGPBについても実行される。走行中の論理プロセッサは、この周期タイマ割り込み発生により、中断されて、スケジューラ4 0 2 により、上記中断処理をされ、サービス量Uが更新され、さらに、更新後のサービス量Uと制御値Bとの比較により、レディキュー4 1 0 か、アウトサービスキュー4 2 0 にキューイングされる。そのあと、上記の制御値Bの設定部4 0 6 の処理が行われる。制御値AとBは、該各区間における制御値を表すものであり、したがって、上記のBの更新が終わった後、Uは0に設定される。この更新が終わった後、スケジューラ4 0 2 は、キュー操作部4 0 5 をコールし、アウトサービスキューの全ての論理プロセッサタスク制御ブロック LGPBをレディキュー4 1 0 にキューイングし、それらの制御ブロック LGPBに対応する論理プロセッサを、次の該実時間の区間において、ディスパッチの対象とする。線4 6 1 がこのデータの動きを表し、線4 6 6 がこのキュー操作の制御の動きを表す。

【0019】この発明の特徴は、制御値B設定部4 0 6 の働きである。ここで、各論理プロセッサのサービス量の上限值である制御値Bを区間毎に変動させることにより、長期観測実時間において、CPUサービス量がA以

8

下となるように制御することができる。たとえばある区間におけるCPUサービス量が現在の制御値B₀(制御値Bの初期値はAである)より、10%少ない場合は、次の該実時間の区間における制御値B₁は、A+10にする。この動的な制御値により、いくつかの区間においては、制御値Aを超えることがあっても、長期観測実時間における平均CPUサービス量をA以下とするという大域的なリソースキャッピングを実現することができる。また、制御値設定部4 0 6 において制御値を常にAとすれば、該各区間全てにおいて、CPUサービス量がA以下となるので、従来の局所的リソースキャッピングを実現することもできる。さらに、VM毎に制御値設定部4 0 6 の働きを変えることにより、VM毎に大域的リソースキャッピングと局所的リソースキャッピングとを別々に実現することもできる。

【0020】図2は、ホスト実計算機システム10を表す。仮想計算機VMは、ホスト実計算機システム10のイメージを持つ。仮想計算機VMの構成図を図3に示す。ホスト実計算機システム10は、一台以上の命令プロセッサを持っている。図2の場合は、4台の命令プロセッサ1-1、1-2、1-3、1-4を示している。命令プロセッサの台数は、これより多くても少なくても良い。各命令プロセッサは、線2-1、2-2、2-3、2-4を経由して、主記憶装置3につながる。仮想計算機システムは、一台以上の仮想計算機VMからなるが、仮想計算機システム全体を制御するプログラムをハイパバイザと呼ぶ。ハイパバイザは、主記憶装置3のある領域9に常駐している。この領域はシステムの管理領域であり、各VMはアクセスできない。ホスト実計算機システムは、外部記憶装置であるディスク装置8-1、8-2、・・・、8-Nを持つ。それらは、ディスク制御装置7に接続されており、その制御下にある。5は、I/Oプロセッサであり、主記憶装置3と線4を経由して接続され、さらに線6を経由してディスク制御装置7に接続される。I/Oプロセッサ5は、主記憶装置3と外部記憶装置(ディスク制御装置7がその例であるが、その他の外部記憶装置でも良い)との間のデータ転送を制御する。

【0021】図3は、仮想計算機VM 20の構成を示す。これは、ホスト実計算機システム10のイメージを持つ。仮想計算機20は、一台以上の論理命令プロセッサ(論理プロセッサと略称する)を持っている。図2の場合は、2台の論理プロセッサ21-1、21-2を示している。論理プロセッサの台数は、これより多くても少なくても良い。各論理プロセッサは、線22-1、22-2を経由して、VMの主記憶領域30につながる。VMの主記憶領域30は、実主記憶装置3を分割した領域によって実現される。または、ハイパバイザが構成する仮想空間によって実現されることもある。論理プロセッサは、命令プロセッサのイメージを持ち、一台の

9

命令プロセッサにくくりつけられる、すなわち、それを専有することにより実現するか、または、他の論理プロセッサとの間で一台の命令プロセッサを実時間を分割して共有するかして実現される。仮想計算機システムは、一台以上の仮想計算機VMからなるが、仮想計算機システム全体を制御するプログラムをハイパバイザと呼ぶ。ハイパバイザは、主記憶装置3のある領域9に常駐している。この領域はシステムの管理領域であり、各VMはアクセスできない。仮想計算機VM 20は、外部記憶装置である論理ディスク装置80-1、80-2、・・・、80-Nを持つ。これらは、実のディスク装置8-1、8-2、・・・、8-Nを専有するか、または、論理プロセッサ間で共有するかによって実現される。それらは、論理ディスク制御装置70に接続されており、その制御下にある。論理ディスク制御装置70は、実ディスク制御装置7を専有するか、または、論理プロセッサ間で共有するかによって実現される。50は、論理I/Oプロセッサであり、VMの主記憶領域30と線40を經由して接続され、さらに線60を經由して論理ディスク制御装置70に接続される。論理I/Oプロセッサ50、実I/Oプロセッサ5を専有または共有することにより実現される。論理I/Oプロセッサ50は、主記憶領域30と外部記憶装置(論理ディスク制御装置70がその例であるが、その他の外部記憶装置でも良い)との間のデータ転送を制御する。以上の仮想計算機VMの実現方式は、従来と同じ方式である。

【0022】図4はVMの定義画面100を表す。101は、VMの名称を定義するフィールドであり、VM1、VM2、VM3、VM4の4台のVMが定義されている。定義台数はこれより多くても少なくても良い。102は各VMのサービス率を定義する部分であり、VM1、VM2、VM3、VM4の各サービス率が、それぞれ、S1、S2、S3、S4であることを示している。このサービス率は、ユーザが指定することができる。103は、各VMの論理プロセッサLGP(p,q)を定義するフィールドである。103の場合、VM1は論理プロセッサLGP(1,1)、LGP(1,2)、LGP(1,3)、LGP(1,4)の4台の論理プロセッサからなる。VM2は論理プロセッサLGP(2,1)、LGP(2,2)、LGP(2,3)の3台の論理プロセッサからなる。VM3は論理プロセッサLGP(3,1)、LGP(3,2)の2台の論理プロセッサからなる。VM4は論理プロセッサLGP(4,1)、LGP(4,2)の2台の論理プロセッサからなる。各VMの論理プロセッサの台数はこれより多くても少なくても良い。このVMの定義画面は従来と同様の方法であるが、108が従来と異なるフィールドであり、各VM毎に大域的なりソースキャッピングを指定する(RC=YG)か、局所的なりソースキャッピングを指定する(RC=YL)か、リソースキャッピングを

10

指定しない(ブランク)かを示す。

【0023】図5について説明する。107は、実時間の分割の方法について表したものである。104は、実時間 τ の流れを表す。106は短い時間 Δt (1秒程度、マイクロ秒 i.e. $\mu \text{ sec}$ 単位)である。 Δt の大きさはシステムパラメタとして変更可能である。実時間 τ は、この短い時間 Δt 106(これを以後区間と呼ぶ)によって分割して考えられる。0の時点がスタート時点を表し、システムの立ち上げ時に、全命令プロセッサ共通に、スタート時点0が設定される。105-1は第1区間である。105-2は第2区間である。105-iは、第i区間である。後述するが、この実時間の分割は全命令プロセッサに共通であり、システム立ち上げ時に、各区間での時間経過毎に、全命令プロセッサで同期してタイマの割込みが発生するように設定される。

【0024】図6は使用する記号の説明である。即ちC(p,q)は、論理プロセッサLGP(p,q)の上記区間 Δt におけるCPUビジータイム(busy time)で単位は、 $\mu \text{ sec}$ である。この値は、当然、区間によって異なるが、同じ記号で表す。 Δt は上記の区間の時間的長さであり、単位は、 $\mu \text{ sec}$ である。A1(%)はVM1のCPU利用率の初期制御値である。A2(%)はVM2のCPU利用率の初期制御値である。AN(%)はVMNのCPU利用率の初期制御値である。A(p,q)(%)は、論理プロセッサLGP(p,q)のCPU利用率の初期制御値である。このA1、A2、・・・、AN、A(p,q), $p=1,2,\dots$; $q=1,2,\dots$ とS1、S2、・・・、SN、との関係は以下の通りである。

【0025】ホスト実計算機システム10における命令プロセッサの台数をK台(Kは1以上)とすると、このK台の命令プロセッサの台数分の全処理能力(P)を、S1、S2、・・・、SNで比例配分した値をA1、A2、・・・、ANとする。このA1、A2、・・・、ANをそれぞれVM1、VM2、・・・、VMNの制御値と呼ぶ。この全処理能力(P)は、100で表しても良いし、100*Kで表しても良い。VMを構成する各論理プロセッサには均等割りした処理能力が与えられる。今、分かり易くするために、1台の実命令プロセッサの処理能力を100として考える。

【0026】各VM_pの制御値A_pに対して、 $p=1,2,3,\dots$

$A(p,q) = A_p / (\text{VM}_p \text{の論理プロセッサの台数})$
 $q=1,2,3,\dots$ とする。これを、論理プロセッサLGP(p,q)の制御値と呼ぶ。

【0027】ただし、A(p,q)が100を越える場合は、以下の補正に従う。その越えた値をD(p,q)とする。A(p,q)=100として、VM_p全体での越えた値D_pは、

$D_p = D(p,1) + D(p,2) + D(p,3) + \dots$

11

となる。この D_p の値を、 VM_p 以外の $\{VM_r\}$ のサービス率 $\{S_r\}$ で比例配分する。このとき、論理プロセッサの制御値が既に100に達している場合は、その VM は比例配分対象から外す。この比例配分した値を、各 $\{VM_r\}$ の制御値 $\{A_r\}$ に加える。こうして更新された制御値 A_r を VM_r の論理プロセッサ台数で割った値を論理プロセッサ $LGP(p, q)$ の制御値とする ($q=1, 2, 3, \dots$)。

【0028】この補正を100を越える各 $A(p, q)$ について繰り返し、対象の VM が無くなったなら終わりとする。10
以上の制御値の計算方法は、従来と類似の方法であるが、補正の方法は独自の方式である。

【0029】図7を説明する。110は、論理プロセッサタスク制御ブロック $LGPB$ を表す。

【0030】ハイパバイザは、論理プロセッサの状態を管理したり、ディスパッチしたり、走行させたりするのに、その主記憶中に論理プロセッサ制御ブロック $LGPB$ を構成する。論理プロセッサ1台に対して一つの制御ブロック $LGPB$ を構成する。これは、 VM をアクティブにしたときに、その各論理プロセッサに対して構成される。20

【0031】制御ブロック $LGPB$ は、状態フラグ111、論理プロセッサ番号112、論理プロセッサリソースデータ113、制御データ114を含む。状態フラグ111は、論理プロセッサの状態 S を含む。115はこの記号の意味を述べる。 $S=0$ のとき論理プロセッサはスリープ(sleep)状態である。 $S=1$ のとき論理プロセッサはレディ(ready)状態である。 $S=2$ のとき論理プロセッサはアウトサービス(out-service)状態である。 $S=3$ のとき論理プロセッサはウエイト(wait)状態である。スリープ状態とは、まだ論理プロセッサがスタートされていない状態である。レディ状態とは、走行可能であるが命令プロセッサが使用中であるために、その割当を待っている状態である。アウトサービス状態とは、論理プロセッサがその制御値に達するCPUサービス量を受けたため、一時的に、ディスパッチ対象から外された状態である。ウエイト状態とは、論理プロセッサが或事象の発生を待っている状態である。この中でディスパッチ対象となるのは、レディ状態の論理プロセッサだけである。112は、 VM 番号 p ($=1, 2, 3, \dots$)とその VM における論理プロセッサ番号 q ($=1, 2, 3, \dots$)を含む。論理プロセッサリソースデータ113は、ゲスト汎用レジスタデータ113-1、ゲスト制御レジスタデータ113-2、ゲストPSWデータ113-3、ゲストCPUタイマデータ113-4を含む。ゲストPSWデータ113-3は、システムマスク SM (ビット0からビット7まで)や、次に実行する命令のアドレス $NI A$ (ビット32から63まで)を含む。システムマスク SM の各ビットは、割り込みの可能性を制御するビットを含む。

【0032】図8は、論理プロセッサタスク制御ブロック $LGPB$ の中に含まれる制御データ114の中のデータ 50

12

を示す。114-1は、論理プロセッサ (p, q) のタイムスライス初期値 $TS_0(p, q)$ (単位は μsec)である。ビット0は符号ビットであり、1のときは負の値を表し、ビット51が、 $1\mu sec$ の位である。タイムスライスの標準値は、システム立ち上げ時に決定されるが、その件は後述する。114-2は、論理プロセッサ (p, q) の残りのタイムスライス $TS(p, q)$ (単位は μsec)である。ビット0は符号ビットであり、1のときは負の値を表し、ビット51が、 $1\mu sec$ の位である。114-3は、論理プロセッサ (p, q) が実時間の区間において受けたCPUサービス量を表すデータ $C(p, q)$ (単位は μsec)を含む。ビット0は符号ビットであり、1のときは負の値を表し、ビット51が、 $1\mu sec$ の位である。これらのデータは、CPUタイマと同じフォーマットである。114-4は、論理プロセッサ (p, q) の初期制御値 $A(p, q)$ (%)を含む。この $A(p, q)$ は、前述の如く図4の102に示す各 VM のサービス率 S_1, S_2, \dots, S_N によって決定される論理プロセッサのサービス量の上限值である。114-5は、論理プロセッサ (p, q) の現在の制御値 $B(p, q)$ (%)を含む。 $B(p, q)$ の初期値は $A(p, q)$ である。114-6は、現在の区間における論理プロセッサ (p, q) のCPU利用率(%) $U(p, q)$ を含む。区間 i の $B(p, q)$ は、 $A(p, q)$ に、直前の区間の差分 $B(p, q) - U(p, q)$ を加えた値によって、求められる。114-7は現在の区間CPU充足度 $D(p, q)$ (%)を表す。 $D(p, q)$ は、現在の区間における制御値 $B(p, q)$ に対する論理プロセッサ (p, q) のCPU利用率の割合、すなわち、

$$D(p, q) = (U(p, q) / B(p, q)) * 100 \quad (\%)$$

である。114-8は統計データ不当インディケータ(1ビットの値 $sinv(p, q)$)であり、 $sinv(p, q)=1$ のときは、上記の114-2($TS(p, q)$)、114-3($C(p, q)$)、114-6($U(p, q)$)、114-7($D(p, q)$)が古いデータである、すなわち、最新のデータではないことを表す。初期値は0である。114-9はリソースキャッピング制御データ rc であり、この論理プロセッサの属する VM がリソースキャッピングの指定無し($rc=0$)か、局所的なりソースキャッピングを指定している($rc=1$)か、大域的なりソースキャッピングを指定している($rc=2$)かを表す。

【0033】図9は、現在走行中の論理プロセッサの制御データを表す。図10は、走行中の論理プロセッサ制御ブロック($LGPB$)のアドレスレジスタ307、及び、その制御ブロック110-R($LGPB-R$)を表す。110-Rは、該当の VM 番号 p 及び論理プロセッサ q を表す。114-Rは、その制御データであり、図9に示す制御データを含む。すなわち、114-1-Rは、走行中の論理プロセッサ (p, q) のタイムスライス初期値 $TS_0_R(p, q)$ (単位は μsec)である。ビット0は符号ビットであり、1のときは負の値を表し、ビット51が、 $1\mu sec$

13

の位である。114-2-Rは、走行中の論理プロセッサ(p,q)の残りのタイムスライスTS_R(p,q)(単位は μ sec)である。ビット0は符号ビットであり、1のときは負の値を表し、ビット51が、1 μ secの位である。114-3-Rは、走行中の論理プロセッサ(p,q)が実時間の区間において受けたCPUサービス量を表すデータC_R(p,q)(単位は μ sec)を含む。ビット0は符号ビットであり、1のときは負の値を表し、ビット51が、1 μ secの位である。これらのデータは、CPUタイマと同じフォーマットである。114-4-Rは、走行中の論理プロセッサ(p,q)の初期制御値A_R(p,q)(%)を含む。このA_R(p,q)は、前述の如く図4の102に示す各VMのサービス率S1、S2、・・・、SNによって決定される論理プロセッサのサービス量の上限值である。114-5-Rは、走行中の論理プロセッサ(p,q)の現在の制御値B_R(p,q)(%)を含む。B_R(p,q)の初期値はA_R(p,q)である。114-6-Rは、現在の区間における走行中の論理プロセッサ(p,q)のCPU利用率U_R(p,q)(%)を含む。区間iのB_R(p,q)は、A_R(p,q)に、直前の区間の差分B_R(p,q)-U_R(p,q)を加えた値によって、求められる。114-7-Rは、走行中の論理プロセッサの現在の区間CPU充足度D_R(p,q)(%)を表す。D_R(p,q)は、現在の区間における制御値B_R(p,q)に対する論理プロセッサ(p,q)のCPU利用率の割合、すなわち、

$$D_R(p,q) = (U_R(p,q) / B_R(p,q)) * 100 \quad (\%)$$

である。114-8-Rは、走行中の論理プロセッサの統計データ不当インディケータ、114-9-Rは、同リソースキャッピング制御データである。

【0034】図11は、タイマ・リクエスト・ブロックTRQB 501を表す。これは、ハイパバイザが一定の時刻にタイマ割込みを発生させたいときに、その主記憶内に構成するものである。502は、その時刻VALを含む。これは64ビットあり、ビット51が1 μ secの単位を表す。これは、ホスト実計算機システムの命令プロセッサの所有している現在時刻タイマ(TODタイマという)と同じ形式である。503は、割込みリターンアドレスであり、IRA(Interrupt Return Address)として表されている。これは、502に示す時刻にタイマ割込みが発生したときに、コントロールを渡すべきアドレスを示している。502及び503の内容はハイパバイザにより設定される。

【0035】図12は、タイマ・リクエスト・キューの構造を表す。これは、ハイパバイザがその主記憶上に構成するソフトウェアのキューである。510は、このキューのヘッダであり、501-1はタイマ・リクエスト・ブロックTRQB-1であり、502-1は、そのタイマ要求値、503-1はその割込みリターンアドレスである。さらに、501-2はタイマ・リクエスト・ブロックTRQB-2であり、502-2は、そのタイマ要求値、5

14

03-2はその割込みリターンアドレスである。501-3はタイマ・リクエスト・ブロックTRQB-3であり、502-3は、そのタイマ要求値、503-3はその割込みリターンアドレスである。この図では3個のタイマ・リクエスト・ブロックTRQBしかキューイングされていないが、その数は動的に変動し、空すなわちヘッダだけのときもある。また、タイマ・リクエスト・ブロックTRQBはそのタイマ要求値VALの小さい値から大きな値への順序でキューイングされる。

【0036】図13はフリー・タイマ・リクエスト・キューを表す。フリー・タイマ・リクエスト・キューは、フリーなすなわち利用可能なタイマ・リクエスト・ブロックTRQBをキューイングしたものであり、システム立ち上げの時に、ハイパバイザにより作られる。501-f-1は、フリーなタイマ・リクエスト・ブロックTRQB-1である。501-f-2は、フリーなタイマ・リクエスト・ブロックTRQB-2である。501-f-3は、フリーなタイマ・リクエスト・ブロックTRQB-3である。このTRQBの個数は動的に変動する。このフリーなTRQBのVAL(502-f-1、502-f-2、502-f-3)及び、IRA(503-f-1、503-f-2、503-f-3)は、実際に使用するときに設定されるので任意の値で良い。

【0037】図14は、システム制御データを表す。500は区間の長さ Δt (単位は μ sec)を含む。これは、図5に示す107実時間の分割におけるひとつの区間の時間的長さである。この区間の時間的長さはシステム立ち上げの時にハイパバイザにより決定されるが、ユーザがシステム・パラメタとして与えることもできる。この500は、システム立ち上げ時にハイパバイザにより、その主記憶内に構成される。これは、ビット0からビット63まで64ビットあり、ビット51が μ secの単位であり、この形式は、ホスト実計算機システムのTODタイマと同じ形式である。501はシステムでのタイムスライスの標準値TS1(μ sec単位)である。

【0038】これは、システム立ち上げの時にハイパバイザにより決定されるが、ユーザがシステム・パラメタとして与えることもできる。この501は、システム立ち上げ時にハイパバイザにより、その主記憶内に構成される。これは、ビット0からビット63まで64ビットあり、ビット51が μ secの単位であり、この形式は、実命令プロセッサのCPUタイマと同じ形式である。550は、スケジューラタイマ同期用のデータ領域である。550-1は1ビットのデータK1であり、命令プロセッサ1のスケジューラ・タイマ設定完了指示データである。550-2(K2)、550-3(K3)、550-4(K4)も同様である。これは、システム立ち上げ時にハイパバイザにより、その主記憶内に、命令プロセッサの台数分確保され、その初期値は0に設定される。

【0039】図15は、システム共通レジスタの現在時

15

刻を表すTODタイマ600を表す。

【0040】これは、ビット0からビット63まで64ビットあり、ビット51が μ secの単位である。これは、従来のTODタイマと同じである。

【0041】図16は、論理プロセッサタスクの状態遷移図を表す。ハイパバイザは、論理プロセッサタスクをその主記憶上の制御ブロック LGPBで表すことは、図7の110に示す通りである。さらに、LGPBは、その中に状態フラグ111を持ち、その値Sは、115に示すとおり
10の意味を持つ。図16において、120は、論理プロセッサタスクがレディ (ready) 状態であることを示している。そのとき該当のLGPBのS=1である。121は、論理プロセッサタスクがウェイト (wait) 状態であることを示している。そのとき該当のLGPBのS=3である。122は、論理プロセッサタスクがアウトサービス (out-service) 状態であることを示している。これは、また、区間におけるサービスを完了している状態、すなわち、該論理プロセッサのCPU利用率がその制御値に達していることを示している。そのとき該当のLGPBのS=2である。
123は、論理プロセッサタスクがスリープ (sleep) 状態であることを示している。そのとき該当のLGPBのS=0
20である。矢印124は、レディ状態からウェイト状態への状態遷移を表す。他の矢印125、126、127、128、129も同様にその起点の状態から終点の状態への状態遷移を表す。

【0042】図17はレディキュー (Ready Queue) 410、アウトサービスキュー (Out Service Queue) 420、ウェイトキュー (Wait Queue) 430からなる。これらは、皆、ハイパバイザによって、その主記憶内に構成されるソフトウェア的な構造体である。200は、レディキューのヘッダであり、410-1、410-2、410-3、410-4 は、レディ状態にある論理プロセッサ制御ブロック LGPB (その状態フラグS=1) であり、順番にキューイングされている。410-1が先頭のLGPBである。201は、レディキューのヘッダであり、420-1、420-2、420-3 は、アウトサービス状態にある論理プロセッサ制御ブロック LGPB (その状態フラグS=2) であり、順番にキューイングされている。420-1が先頭のLGPBである。202は、ウェイトキューのヘッダであり、430-1、430-2、430-3、430-4 は、ウェイト状態にある
40論理プロセッサ制御ブロック LGPB (その状態フラグS=3) であり、順番にキューイングされている。430-1が先頭のLGPBである。どのキューにおいてもそのキューイングされているLGPBの個数すなわちキューの長さは可変であり、論理プロセッサ実行中に動的に変動する。これらのキューのヘッダはシステム立ち上げ時に、ハイパバイザにより、作られ初期化されるが、キューの実体 (LGPBがキューイングされている構造) は、ハイパバイザのスケジューラにより、作成・管理される。

16

【0043】図18は命令プロセッサの所有するレジスタを表す。これらのレジスタは各命令プロセッサ毎にある。300は、ホストCPUタイマレジスタであり、ビット0からビット63の64ビットから成り、ビット51が 1μ secの単位である。これは従来のCPUタイマと同一であり、ビット51から 1μ sec毎に1引かれ、ビット0が1即ち、値が負の時割込み (これをCPUタイマ割込みという) がハードウェア的に発生する。301は、ゲストCPUタイマレジスタであり、ビット0から
10ビット63の64ビットから成り、ビット51が 1μ secの単位である。これは、この命令プロセッサで走行中の論理プロセッサのCPUタイマとして与えられるものであり、従来のCPUタイマと同様の動作を、走行中の論理プロセッサに対して行う。302は、ホスト・クロック・コンパレータであり、従来と同一の仕様である。すなわち、ビット0からビット63の64ビットから成り、ビット51が 1μ secの単位である。この値は、一定値であり、符号無し
20の2進数として、図15の600の現在時刻TODタイマの値とハードウェアにより、比較される。TODタイマ600の値が、クロックコンパレータ302の値を越えるとハードウェア的に割込みが発生する。303はゲスト汎用レジスタであり、304はゲスト制御レジスタであり、305はゲストPSWレジスタである。305は、走行中の論理プロセッサのPSWとして動作するものであり、実命令プロセッサのPSWと同様の働きを行う。すなわち、ビット0から7は論理プロセッサのシステムマスク SMであり、ビット32から63は次に実行する論理プロセッサの命令のアドレス NIAを表す。

【0044】図19はモード・インディケータ305を表す。305は1ビットからなり、その値Iは、306に示す意味を持つ。すなわち、I=1のとき、命令プロセッサは、VMモードであり、I=0のとき命令プロセッサは、基本モードであることを表す。これは、ハードウェア的なインディケータである。

【0045】以上の種々のハードウェアリソース、ソフトウェアリソースをベースにして、本発明におけるシステムの動作方法をフローチャートを用いて以下に説明する。

【0046】図20に、システムの立ち上げすなわちハイパバイザの立ち上げ時の初期化方法について述べる。ハイパバイザ立ち上げ時、図13に示すフリーTRQBキューを構成する。その主記憶内にTRQBを構成しヘッダ520からキューイングする (720)。TRQBの個数は最大サポートVM台数* (VM当たりの最大論理プロセッサ台数) 分用意する。例えば、10台のVMと各VMについて論理プロセッサの台数10台であれば、 $10 * 10 = 100$ 個のTRQBを用意して、キューイングしておく。各TRQBの内容は0クリアしておく。次に図14に示すシステム・タイム・スライスTS1を計算し、システムの
50

17

領域501に格納する(721)。システム・タイム・スライスは、いろいろな決定方法があるが、ここでは、1,000ステップだけ単純命令(例えばLoad命令)を実行し、その実行時間をシステム・タイム・スライスTS1(単位は μ sec)とすることにする。次に図14に示す実行時間を分割する区間の長さ Δt (単位は μ sec)を決定し、システム領域500に格納する(722)。 Δt の値は、ここでは1秒すなわち、1,000,000 μ secとするが、この値は、ユーザ指定のシステム・パラメタにより変更可能とする。次に図12に示すタイマリクエストキューのヘッダ510を構成し、それを初期化し、キューの中に何もキューイングされていない状態、すなわち、空の状態にする(723)。

【0047】次に図14に示すスケジューラタイマ同期データ領域550を0クリアする。すなわち、550-1のK1、550-2のK2、550-3のK3、550-4のK4を0クリアする。この例では4台の命令プロセッサの場合であるが、N台の場合は、K1,K2,...,KNまでのデータを0クリアする(724)。次に、このデータを用いてスケジューラタイマの同期処理を行う(725)。これで、立ち上げ時の初期化を終える。

【0048】この725の処理について、図21に詳述する。まず、図13のフリーTROBキューの先頭のTROBをキューより取り外し、次のTROBを先頭とする。該TROBのタイマ要求値(502相当)に以下の式によって計算されるVALOを格納する(725-1)。

【0049】

$VALO = \text{現在時刻}TOD(600) + \text{区間の長さ} \Delta t(500)$
 該TROBの割り込みリターン・アドレスIRAにスケジューラのある入り口アドレス(E0)を設定する(725-2)。この入り口アドレス(E0)は、ハイババイザによって決められるアドレスである。該TROBを図12のタイマ・リクエスト・キューにキューイングする(725-3)。このときタイマ・リクエスト・キューは空の状態であるから、このTROBは、先頭にキューイングされる。次に該TROBの上記のタイマ要求値VALOを現命令プロセッサのホスト・クロック・コンパレータ・レジスタ302に設定する。さらに、現命令プロセッサの番号をiとすると、

$550-i(\text{値}K_i)=1$

とする(725-4)。これにより、現命令プロセッサについては、現在時刻から $\Delta t \mu$ sec後に、ホスト・クロック・コンパレータの割り込みが発生するようになる。次に他の命令プロセッサにも同様のタイマを設定するために上記のVALOをパラメタとしてプロセッサ間スケジューラタイマ同期処理を行う(730)。以上でスケジューラタイマ同期処理を終える。

【0050】この730の処理を図22に詳述する。すなわち、タイマ要求値VALOをパラメタとして、自分以外の全ての命令プロセッサに対して外部割り込み要求を発

18

行する。このとき、これが、スケジューラタイマ同期要求であることを示すパラメタも、相手命令プロセッサへ渡す(730-2)。次にスケジューラタイマ同期データ領域550の全てのK1,K2,...,KNについて、K1=1かつK2=1かつ...かつKN=1であるかをチェックする(730-3)。これらのデータの初期値は0であり、命令プロセッサ(番号i)が、タイマ要求値VALOに対して、そのクロック・コンパレータの設定を完了したとき、すなわち、図21に示す725-4相当の処理を完了したとき、550のKi=1とする。したがって、730-3は、全ての命令プロセッサが、そのクロック・コンパレータに対して上記のタイマ要求値VALOを設定するのを待つものである。この完了後図20の726へ行く(すなわち、ハイババイザの立ち上げ処理を終える)。

【0051】図23に外部割り込み処理を示す。図22に示すプロセッサ間スケジューラタイマ同期処理で、各プロセッサにスケジューラタイマ同期要求の外部割り込み要求を発行する(730-2)と、各プロセッサに外部割り込みが発生する。図23は、一般的な外部割り込みの処理を示す。まず、外部割り込みの種類を判断する。すなわち、スケジューラタイマ同期要求の外部割り込みかを判断する(740)。これは、従来どおり、送られてきた割り込み要求コードから判断できる。そうではないときは、クロック・コンパレータ割り込みかどうかを判断する(740-1)。そのときは、クロック・コンパレータ割り込みの処理を行うために910-1へ行く。そうではないときは、従来通り、該当の外部割り込み処理を行い、一般的なスケジューラ999へ行く(741)。そうであるときは、渡されたパラメタVALOを現プロセッサのホストクロックコンパレータ302に設定する(742)。これにより、ほぼ、時刻VALOに、現命令プロセッサのホスト・クロック・コンパレータの割り込みが発生するようになる。このタイマ割り込み要求が、全ての命令プロセッサにおいて設定されたことを確認するために、スケジューラ・タイマ同期データ領域550の全てのK1,K2,...,KNについて、K1=1かつK2=1かつ...かつKN=1であることをチェックする。そうでないときは、くりかえし、同条件をチェックする。そうなったときに、ループから抜けて、一般的なスケジューラ999へ行く(744)。

【0052】これにより、全命令プロセッサにおいて、ほぼ、時刻VALOにホスト・クロック・コンパレータの割り込みが発生することになる。

【0053】ユーザがVMをアクティブ(起動)すると、図24に示すようにVM下の全ての命令プロセッサのタスク制御ブロックLGPBを初期化する(700)。この初期化処理を図25に詳述する。起動したVMの番号をp(p=1,2,...のいずれかの値)とする。VMp下の各論理プロセッサ制御ブロックLGPB(p,q)(q=1,2,3,...)について、以下の初期化を行う。各LGPBは、図7

19

に示す論理プロセッサ・タスク制御ブロック110の構造を持つ。すなわち、タイムスライス初期値 $TS_0(p,q)$ を計算し、該LGPBの114-1のところに格納する。この $TS_0(p,q)$ は、システム・タイム・スライス標準値 $TS_1(501)$ を、そのまま持って来ても良い。次に残りのタイムスライス $TS(p,q)$ に、 $TS_0(p,q)$ を初期値として設定し、114-2に格納する。次に区間CPUサービス量データ $C(p,q)$ を0に初期化して114-3に格納する。初期制御値 $A(p,q)$ を計算し、114-4に格納する。この計算方法は、図4、図6で説明したとおりアクティブなVMの間のサービス率に基づいて計算される。次に制御値 $B(p,q)$ に $A(p,q)$ を初期値として設定し、114-5に格納する。次に区間CPU利用率 $U(p,q)$ を0に初期化する。次に、区間CPU充足度 $D(p,q)$ を0に初期化する。さらに統計データ不当インディケータ114-8 $\sin v(p,q)=0$ とする。これは、上記の114-2($TS(p,q)$)、114-3($C(p,q)$)、114-6($U(p,q)$)、114-7($D(p,q)$)が最新のデータであることを表す。VM下のすべての論理プロセッサの制御ブロックLGPB(p,q)について、以上の初期化を終えたら700-3へ行き、VMのアクティベイト処理を終える。

【0054】図26の説明。VMのアクティベイト処理を完了すると、そのVM上でオペレーティング・システム(OS)を起動する。そうすると、VM下の全ての論理プロセッサがスタートされてレディキュー410にキューイングされ、コントロールが図26に示すディスパッチ処理に渡される。このディスパッチ処理では、まず、図17のレディキュー410の先頭のタスク制御ブロックLGPBをキューからはずして、取り出す。そのLGPBの112の場所からVM番号p、論理プロセッサ番号qを得る(713-2)。次に、該LGPBの残りのタイム・スライス・データ $TS(p,q)$ を場所114-2から読み出して、現命令プロセッサのホストCPUタイマ・レジスタ300に設定する(713-3)。これにより、もし、この $TS(p,q)$ を該論理プロセッサが使い尽くしたときには、ホストCPUタイマの割込みが発生する。これにより、別の論理プロセッサに制御を移すことができるようになる。次に統計データ不当インディケータ $\sin v(p,q)=1$ とする(713-4)。これは、114-2($TS(p,q)$)、114-3($C(p,q)$)、114-6($U(p,q)$)、114-7($D(p,q)$)が古いデータである、すなわち、最新のデータではないことを表す。これは、これからまさに該論理プロセッサ(p,q)をディスパッチしようとしており、ディスパッチ後はこれらのデータは最新でなくなるからである。さらに、該LGPB内の論理プロセッサリソースデータ113を現在の命令プロセッサのレジスタに設定し、該論理プロセッサを、現命令プロセッサ上で走行させる。すなわち、該論理プロセッサのLGPB内の汎用レジスタデータ113-1をゲスト汎用レジスタ303に設定し、制御レジスタデータ113-2をゲスト制御レジスタ3

20

04に設定し、PSWデータ113-3をゲストPSWレジスタ305に設定し、CPUタイマ・データ113-4をゲストCPUタイマ301に設定し、動作モードをVMモードにする。即ち、モードレジスタ305のI=1とし、さらに、該LGPBのアドレスを、走行中論理プロセッサ表示レジスタ307に設定し、コントロールをゲストPSWレジスタ305が示すところへ渡す。以後命令プロセッサは、ゲストPSWレジスタ305、及びゲスト制御レジスタ304によって制御される。すなわち、該当の論理プロセッサを現命令プロセッサ上で走行させることになる(713-5)。以上によってディスパッチ処理を終える。

【0055】図27、28、29の説明：このようにして、論理プロセッサが実際の命令プロセッサ上で走行させられる。走行中の論理プロセッサは、いろいろな要因で中断される。そのひとつは、ホストの割込みである。たとえば、ホストのクロック・コンパレータの割込みや、ホストのCPUタイマの割込みがそうである。ホスト・クロック・コンパレータ割込みは、図18に示す、ホスト・クロック・コンパレータ・レジスタ302の値が、図15に示すシステム共通レジスタ600の現在時刻TODの値を越えたときに発生する。これは、従来通りである。ホストCPUタイマの割込みは、図18に示すホストCPUタイマ・レジスタ300の値が負になったときに発生する。これも従来通りである。図21に示すスケジューラタイマ同期処理により、全命令プロセッサで同時にスケジューラタイマ用のホスト・クロック・コンパレータの割込みが発生する。また、図26に示す、ディスパッチ処理でホストCPUタイマ・レジスタ300に残りのタイム・スライス $TS(p,q)$ (論理プロセッサ(p,q)用のタイムスライス)を設定したことにより(713-3)、該当の論理プロセッサが、そのタイム・スライスを使い切るとホストCPUタイマ割込みが発生する。これを、タイムスライスエンドの割込みという。これらの制御は、従来通りである。これらのホスト割込み処理の後、図27、28、29に示す論理プロセッサの中断処理が呼ばれる。ここでは、まず、走行中であった論理プロセッサの制御ブロックLGPBのアドレスをレジスタ307から求める。今それをLGPB_Rとする(800-2)。このLGPB_Rから現VM番号p、論理プロセッサ番号qを求める(800-3)。これは、LGPB_Rの場所112から読み出してくればよい(図7参照)。次に、LGPB_Rの統計データ不当インディケータ $\sin v(p,q)=1$ であるかを判断する。そうである場合は、該LGPB_R内の種々の統計データを更新する必要がある。800-4へ行く。そうでない場合は、すでに更新済みであるので、ただちにリターンする。次に、該論理プロセッサのCPU消費時間を計算するために、現命令プロセッサのホストCPUタイマレジスタ300の値を読み出す。今、その値をTS_Rとする(800-4)。次に該LGPB_Rの残

21

りのタイムスライスデータを114-2-Rから読み出す。今、その値をTS_Rとする(800-5)。これは、該論理プロセッサのディスパッチ時のホストCPUタイマの値がTS_R μ secであり、それが1 μ secごとにビット51から1が減算されて行き(図18の300参照)、今処理している論理プロセッサの中断時にTS_R' μ secとなったことを意味する。したがって、該論理プロセッサのCPU消費時間u μ secは、

$$u = (TS_R) - (TS_{R'})$$

によって求められる(800-6)。このu μ secは、最後のディスパッチから、最初の中断までの論理プロセッサの消費したCPU時間である。次に、該論理プロセッサ(p,q)の累計のCPU消費時間を計算する。これは、以下の式による。

【0056】すなわち、LGPB_Rの場所114-3-Rの値C(p,q)_Rを読み出す。これは、該論理プロセッサの区間におけるCPU消費時間の累計を表す。そこで、

$$C(p,q)_R = C(p,q)_R + u$$

とし、更新されたC(p,q)_Rの値を該LGPB_Rの114-3-Rに格納する(800-7)。次に、該論理プロセッサの区間におけるCPU利用率U(p,q)_R(%)を求める。システムデータ領域500から区間の長さ Δt μ secを得る。さらに、800-7で更新したC(p,q)_R(これは、区間における該論理プロセッサのCPU消費時間)を用いて、

$$U(p,q)_R = (C(p,q)_R / \Delta t) * 100 \quad (\%)$$

これを、該論理プロセッサの区間CPU利用率として、該LGPB_Rの114-6-Rに格納する(800-8)。次に、この区間CPU利用率が制御値を越えるかどうかを判断する。すなわち、800-8で計算した区間CPU利用率U(p,q)_R(これは、該論理プロセッサのCPU利用率)と、該LGPB_Rの場所114-5-Rに格納されている制御値B(p,q)_R(これも該論理プロセッサに関する値)を用いて、

$$(U(p,q)_R) < (B(p,q)_R)$$

かどうかを判断する(800-9)。もし、そうであるならば、まだ、現区間においては、制御値に達していないので、再びレディキューにキューイングしてディスパッチするために800-10へ行く。もし、そうでないときは、すでに制御値に達したのでアウトサービスキューへキューイングするために800-12へ行く。先ず、800-10から説明する。ここでは、区間における該論理プロセッサ(p,q)のCPU充足度D(p,q)_R(%)を計算する。

【0057】

$$D(p,q)_R = ((U(p,q)_R) / (B(p,q)_R)) * 100$$

この値を該論理プロセッサ(p,q)の区間CPU充足度として該LGPB_Rの場所114-7-Rに格納する(800-10)。次に、該LGPB_Rをレディキューにキューイングするために900へ行く。900では、レディキュー

22

410に該LGPB_Rをキューイングするために、そのヘッダ200から検索する。このレディキューは、区間CPU充足度D(p,q)の小から大へソートしてキューイングしているので、該LGPB_Rの区間CPU充足度D(p,q)_Rについてソートし適切な位置にキューイングする(900)。レディキューは先頭のLGPBからディスパッチされるので、このソートにより、各論理プロセッサの区間CPU充足度が小さいものから先にディスパッチされることになり、従って、各論理プロセッサ(p,q)の現在の制御値B(p,q)に比例したCPUサービスが行われるようになる。900の処理完了の後は、該LGPB_Rの以上の統計データの更新を終えたので、その統計データ不当インディケータ114-8の値sinv(p,q)=0とする(800-11)。これで中断処理を終えて、呼び出し元(caller)へもどる。次に、区間CPU利用率が制御値に達したときの処理を800-12から説明する。ここでは、該LGPB_Rをアウトサービスキュー420のヘッダ201を検索し、該LGPB_Rをキューの最後へキューイングする(800-12)。この完了後800-11へ行き、該論理プロセッサの統計データ不当インディケータ114-8の値sinv(p,q)=0とする。これにより、中断処理を終え、呼び出しもとへリターンする。

【0058】図30、31にレディキューへのキューイング方法を詳述する。これは900の処理の詳細である。先ず900-2では、レディキュー410をそのヘッダ200より検索し、先頭のLGPBにアクセスし、これをカレントLGPBとする。キューイングすべき論理プロセッサ制御ブロックLGPBを該当LGPBとする(900-2)。該当LGPBの区間CPU充足度(114-7相当部分の値)をD(p,q)_Cとする。カレントLGPBの区間CPU充足度(114-7相当部分の値)をD(p,q)_Gとする。そこで、

$$(D(p,q)_G) \geq (D(p,q)_C)$$

かどうかを判断する(900-3)。そうであるときは、まだ、後方へキューイングする必要があるのであるので、キュー内の次のLGPBが存在するかを判断する(900-4)。あればそれをカレントLGPBとする(900-5)。その後、再び900-3へ行く。900-3の条件が成立しないときは、カレント位置の直前に該LGPBをキューイングする(900-6)。それでキューイングは完了する。

【0059】900-4の判断でキュー内の次のLGPBが存在しないときは、900-7に行き、そこで、レディキューの最後にキューイングする(900-7)。これにより、キューイングを完了する。

【0060】図32、33にクロック・コンパレータ割り込みの処理を述べる。クロック・コンパレータ割り込みは外部割り込みの一種であり、図23に示す外部割り込み処理でクロック・コンパレータ割り込みと判断されると910-1に制御が来る。910-2で、図12のタイマ・リクエスト・キューのヘッダ510より、その

23

先頭のTROBにアクセスする。もし、タイマ・リクエスト・キューが空のときは、910-8へ行く。該タイマ・リクエスト・キューが空でないときは先頭のTROBにアクセスする。そのTROBのタイマ要求値VAL(502)と現在時刻TOD(600)を比較する(910-3)。もし、

VAL < TOD

ならば、該TROBをタイマ・リクエスト・キューより取り外す。これを該当TROBとする(910-4)。かかるTROBが無いときは、910-8へ行く。かかるTROBがあるときは、次のTROBを先頭のTROBするようにヘッダ510を更新し(910-4)、910-5へ行く。910-5においては、該当TROBが、スケジューラ・タイマ割込み用のものであるかを判断する。それは、その割込みリターンアドレスIRAがスケジューラの特定の入り口EOであるかを判断すればよい。特定のアドレスEOを入り口とするスケジューラの処理においては、スケジューラタイマの割込み処理を行う。これは、図34で説明する。スケジューラ・タイマ割込み用のものである場合は、次の区間 $\Delta t \mu \text{sec}$ 後のスケジューラタイマの割込みを発生させるために、該TROBのタイマ要求値VAL(502)に以下の値を設定する(910-6)。

【0061】VAL= 現在時刻TOD(600) + Δt (500)

その後、再び、該TROBを図12のタイマ・リクエスト・キューにキューイングする。このタイマ・リクエスト・キューにおいては、各TROBのタイマ要求値VALの小さい値から大きい値へとソートしてキューイングする(910-7)。このあと910-8へ行く。910-5でスケジューラ・タイマ割込み用のものでないときは、910-8へ行く。910-8では、あらたにタイマ・リクエスト・キューが非空かを判断し、非空のときは、その新しい先頭のTROBのタイマ要求値(502相当部分の値)VALをホスト・クロック・コンパレータ・レジスタ302に設定し(910-9)、910-11へ行く。かかる新しい先頭のTROBが存在しないとき、すなわち、タイマ・リクエスト・キューが空になったときは、現時刻に500ミリ秒程度を加算した値を現命令プロセッサのホスト・クロック・コンパレータに設定する(910-10)。このあと走行中論理プロセッサの中断処理を行うために910-11へ行く。910-11では、図27の処理(走行中論理プロセッサの中断処理)をコールする。その後910-12へ行く。ここでは、910-3と910-4で見つけた該当TROBが存在するときは、その割り込みリターンアドレス(503相当部分の値IRAが、このリターンアドレスを表す)へコントロールを渡す(910-13)。かかるTROBが存在しないときは、一般的なスケジューラ999へ行く(910-14)。

【0062】図34の説明。図34は、スケジューラの特定の入り口EOからの処理を表す。ここでは、スケジュー

24

ーラ・タイマ・割込み処理を行う。ここには、スケジューラ・タイマ用のTROBの割込みリターンアドレスIRAからコントロールが渡ってくる(図33の910-13参照)。このスケジューラ・タイマ割込みは、図21に示すスケジューラタイマ同期処理及び図32に示すホスト・クロック・コンパレータ割込み処理での各区間 Δt 経過後の割込み要求の設定(910-6、910-7)によって、各命令プロセッサで $\Delta t \mu \text{sec}$ 後に同期して発生するようになる。各命令プロセッサは、スケジューラタイマ割込みによって図34に示す処理を行うようになる。すなわち、まず、レディキュー410のスケジューラタイマ割込み処理を行う(920)。そのあと、アウトサービスキュー420のスケジューラタイマ割込み処理を行う(930)。そのあと、ウェイトキュー430のスケジューラタイマ割込み処理を行う(940)。この後一般的なスケジューラの処理999へ行く。

【0063】図35の説明：図35は、レディキューに対するスケジューラタイマ割込み処理、すなわち、920の処理の詳細である。まず、図17のレディキュー410をヘッダ200より、検索し、レディキューが非空であるかを判断し、非空のときは、その先頭のLGPBを該当のLGPBとする。その該当のLGPBのVM番号をp、論理プロセッサ番号をqとする(920-1)。その該当LGPBが無いときは次の処理930へ行く。その該当LGPBが存在するときは、920-4へ行き、該LGPBについて現区間での終了処理を行う。920-4ではまず該LGPBの残りのタイムスライスTS(p,q)(114-2)をタイムスライス初期値TS0(p,q)(114-1)に初期化する。これは、次の区間 Δt においては、タイムスライス初期値から始めれば良いからである。次に該当LGPBの区間CPUサービス量C(p,q)を0クリアする。これも次の区間に向けての初期化である。次に、区間におけるCPUサービス量の制御値B(p,q)を更新する。この制御値の更新を行うことが本特許の特徴的処理である。すなわち、該LGPBのリソースキャッピング制御データ(114-9)rc=2、すなわち、大域的リソースキャッピング指定のとき、該当LGPBの現在の制御値B(p,q)(114-5)、その初期制御値A(p,q)(114-4)、区間CPU利用率をU(p,q)(114-6)とするとき、

$$B(p,q) = A(p,q) + (B(p,q) - U(p,q))$$

とする。この更新されたB(p,q)を該当LGPBの114-5に格納する。これは、区間における未使用の部分(B(p,q)-U(p,q))だけ、初期制御値A(p,q)に加算することにより、大極的に見て、論理プロセッサ(p,q)のCPU利用率をA(p,q)以下に抑制するためである。上記のrc=0or1のときは、リソースキャッピングの指定無しか、局所的なリソースキャッピングの指定であるのでB(p,q)の値は常にA(p,q)に等しくする。これにより、制御値の更新が終わったので、上記区間CPU利用率U(p,q)を0クリアする。次に該当LGPBの区間CPU充足度D(p,q)(114-

7) も OK リアする。これも次の区間に向けての初期化である。以上で該 LGPB の次の区間 Δt に向けての初期化を終える。次にレディキュー上に次の LGPB がキューイングされているかを判断し、あればそれを新たな該当 LGPB とし (9 2 0 - 5)、9 2 0 - 2 へ行く。ここで、再び新たな該当 LGPB が有るかどうかを判定し、有れば、再び、新たな該当 LGPB の初期化のために 9 2 0 - 4 へ行く。なければ、以上でレディキューに対するスケジューラタイマ処理を終えて、次の処理 9 3 0 へ行く。

【 0 0 6 4 】図 3 6 の説明: 図 3 6 は、図 3 4 の処理 9 3 0 の詳細である。先ず、図 1 7 のアウトサービスキュー 4 2 0 をヘッダ 2 0 1 より検索し (9 3 0 - 1)、アウト・サービス・キューが非空であるかを判断し、非空のときは、その先頭の LGPB を該当の LGPB とする。その該当の LGPB の VM 番号を p 、論理プロセッサ番号を q とする (9 3 0 - 2)。その該当 LGPB が無いときは次の処理 9 4 0 へ行く。その該当 LGPB が存在するときは、9 3 0 - 4 へ行き、該 LGPB について現区間での終了処理を行う。これは、図 3 1 の 9 2 0 - 4 と同じ処理である (9 3 0 - 4)。この後該 LGPB をレディキュー 4 1 0 にキューイングする。すなわち、そのヘッダ 2 0 0 から検索して最後の位置にキューイングする (9 3 0 - 5)。これは、次の区間 Δt において、アウトサービスキューに有る全ての論理プロセッサ (p, q) を再び、新制御値 $B(p, q)$ に基づいて、サービスするためである。次にアウトサービスキュー 4 2 0 上に次の LGPB がキューイングされているかを判断し、あればそれを新たな該当 LGPB とし (9 3 0 - 6)、9 3 0 - 3 へ行く。ここで、再び新たな該当 LGPB が有るかどうかを判定し、有れば、再び、新たな該当 LGPB の初期化のために 9 3 0 - 4 へ行く。なければ、以上でアウトサービスキューに対するスケジューラタイマ処理を終えて、次の処理 9 4 0 へ行く。

【 0 0 6 5 】図 3 7 の説明: 図 3 7 は、図 3 4 の処理 9 4 0 の詳細である。先ず、図 1 7 のウェイトキュー (wait queue) 4 3 0 をヘッダ 2 0 2 より検索し (9 4 0 - 1)、ウェイトキューが非空であるかを判断し、非空のときは、その先頭の LGPB を該当の LGPB とする。その該当の LGPB の VM 番号を p 、論理プロセッサ番号を q とする (9 4 0 - 1)。その該当 LGPB が無いときは、処理完了なので一般的なスケジューラ 9 9 9 へ行く。その該当 LGPB が存在するときは、9 4 0 - 3 へ行き、該 LGPB について現区間での終了処理を行う。これは、図 3 5 の 9 2 0 - 4 と同じ処理である (9 4 0 - 3)。次にウェイトキュー 4 3 0 上に次の LGPB がキューイングされているかを判断し、あればそれを新たな該当 LGPB とし (9 4 0 - 4)、9 4 0 - 2 へ行く。ここで、再び新たな該当 LGPB が有るかどうかを判定し、有れば、再び、新たな該当 LGPB の初期化のために 9 4 0 - 3 へ行く。なければ、以上でウェイト・キューに対するスケジューラタイマ処理を終えて、一般的なスケジューラ 9 9 9 へ行く。

【 0 0 6 6 】図 3 8 は一般的なスケジューラの処理である。ここでは、種々のシステムアクティビティの処理を行ったり、統計量を採ったりする。さらにレディキューを検索し、走行可能な論理プロセッサがあればそれをディスパッチしたりする。レディキューが空のときは、アウトサービスキューを検索し、リソースキャッピングの指定のない LGPB、すなわち、その LGPB のリソースキャッピング制御データ (114-9) $rc=0$ の LGPB だけをレディキューに上げてディスパッチする。この処理は従来の一般的なスケジューラの処理と同じであり、詳細は省略する。

【 0 0 6 7 】以上の実現方法によって大域的なリソースキャッピングを実現することができるが、さらに、この実現方式は、従来の局所的なリソースキャッピングも容易に実現することができる。それは、図 3 5 の 9 2 0 - 4 の処理において、該当 LGPB (p, q) の制御値 $B(p, q)$ の更新を止めていつも初期値 $A(p, q)$ とすれば、各区間において CPU サービス量 $U(p, q)$ が、いつも初期値 $A(p, q)$ によって抑制されることになる。これを該当の VM の全論理プロセッサについて行えば、その VM に対して、局所的なリソースキャッピングを実現することができる。これを VM 毎に制御する、すなわち、ある VM は大域的なリソースキャッピング制御を行い、ある VM は局所的なリソースキャッピング制御を行うと云うことも容易である。

【 0 0 6 8 】最後に図 1 のブロック図と、フローチャートとの対応関係を示す。図 1 のシステム初期化 4 0 0 は、図 2 0 のハイパバイザ立ち上げ時の初期化処理を全て含む。

【 0 0 6 9 】周期タイマ割込み発生部 4 0 3 は、図 2 1 のスケジューラタイマ同期処理、図 2 2 のプロセッサ間スケジューラタイマ同期処理、図 3 2、3 3 のホストクロック・コンパレータ割込み処理を全て含む。スケジューラ 4 0 2 は、図 2 3 の外部割込み処理、図 2 4、図 2 5 の VM のアクティベイト処理、図 2 7、2 8、2 9 の論理プロセッサ中断処理、図 3 0、3 1 のレディキューへのキューイング処理、図 3 4 のスケジューラタイマ割込み処理、図 3 5、図 3 6、図 3 7 のレディキュー、アウトサービスキュー、ウェイトキューに対するスケジューラタイマ割込み処理、図 3 8 の一般的なスケジューラの処理の全てを含む。ディスパッチャ 4 0 1 は、図 2 6 のディスパッチ処理を全て含む。4 0 4 サービス量 (U) 計算部分は、図 2 7、2 8、2 9 の論理プロセッサの中断処理に相当する。4 0 5 のキュー操作部分は、図 3 0、3 1 のレディキューへのキューイング処理を全て含む。4 0 6 の制御値 (B) 設定部分は、図 3 5、図 3 6、図 3 7 のレディキュー、アウトサービスキュー、ウェイトキューに対するスケジューラタイマ割込み処理に相当する。

【 0 0 7 0 】

【発明の効果】本特許により、仮想計算機に対するCPUサービス量をユーザの指定量に抑えるというリソースキャッピングの目的において、局所的のみならず大域的に、その目的を実現することができる。仮想計算機毎に局所的または大域的の指定を行うことができる。これにより、ユーザの意図にそったリソースキャッピングを実現することができる。

【図面の簡単な説明】

- 【図1】本特許の概要を表すハイパバイザのブロック、およびコントロールやデータの流れを表す図である。 10
- 【図2】ホスト実計算機システムを表す図である。
- 【図3】仮想計算機の構成図である。
- 【図4】仮想計算機VMの定義画面を表す図である。
- 【図5】実時間の分割方法を示す図である。
- 【図6】記号の説明を表す図である。
- 【図7】論理プロセッサ・タスク制御ブロックの構造を表す図である。
- 【図8】論理プロセッサ(p,q)の制御データを表す図である。
- 【図9】走行中の論理プロセッサ(p,q)の制御データを 20 表す図である。
- 【図10】走行中の論理プロセッサ指示レジスタを表す図である。
- 【図11】タイマ・リクエスト・ブロック(TRQB)の構造を表す図である。
- 【図12】タイマ・リクエスト・キューを表す図である。
- 【図13】フリーTRQBのキューを表す図である。
- 【図14】システムにおける主記憶上の共通データを表す図である。 30
- 【図15】システムにおける共通レジスタを表す図である。
- 【図16】論理プロセッサタスクの状態遷移図である。
- 【図17】レディキュー(ready queue)、アウトサービスキュー(out service queue)、ウェイトキュー(wait queue)を表す図である。
- 【図18】命令プロセッサのレジスタ類を表す図である。
- 【図19】モードレジスタを表す図である。
- 【図20】ハイパバイザ立ち上げ時の初期化処理のフローチャートである。 40
- 【図21】スケジューラタイマに対する同期処理のフローチャートである。
- 【図22】命令プロセッサ間のスケジューラタイマに関する同期処理を表すフローチャートである。
- 【図23】外部割込み処理を表すフローチャートである。

【図24】VMのアクティベイト処理のフローチャートである。

【図25】VMのアクティベイト処理のフローチャートである。

【図26】論理プロセッサに対するディスパッチ処理のフローチャートである。

【図27】論理プロセッサの中断処理のフローチャート(1/3)である。

【図28】論理プロセッサの中断処理のフローチャート(2/3)である。

【図29】論理プロセッサの中断処理のフローチャート(3/3)である。

【図30】レディキューへのキューイング処理のフローチャート(1/2)である。

【図31】レディキューへのキューイング処理のフローチャート(2/2)である。

【図32】ホストクロック・コンパレータ割込み処理のフローチャート(1/2)である。

【図33】ホストクロック・コンパレータ割込み処理のフローチャート(2/2)である。

【図34】スケジューラタイマ割込み処理のフローチャートである。

【図35】レディキューに対するスケジューラタイマ割込み処理のフローチャートである。

【図36】アウトサービスキューに対するスケジューラタイマ割込み処理のフローチャートである。

【図37】ウェイトキューに対するスケジューラタイマ割込み処理のフローチャートである。

【図38】一般的なスケジューラの処理を表す図である。

【図39】あるVMのある負荷に対する本来のCPU利用率を実時間の経過とともに表す図である。

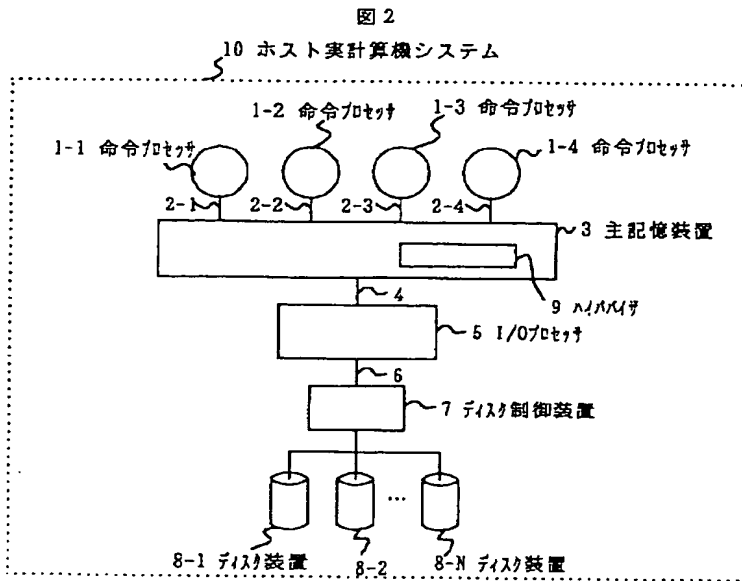
【図40】図39のVMに対して局所的リソースキャッピングを適用したときのCPU利用率を表す図である。

【図41】図39のVMに対して大域的リソースキャッピングを適用したときのCPU利用率を表す図である。

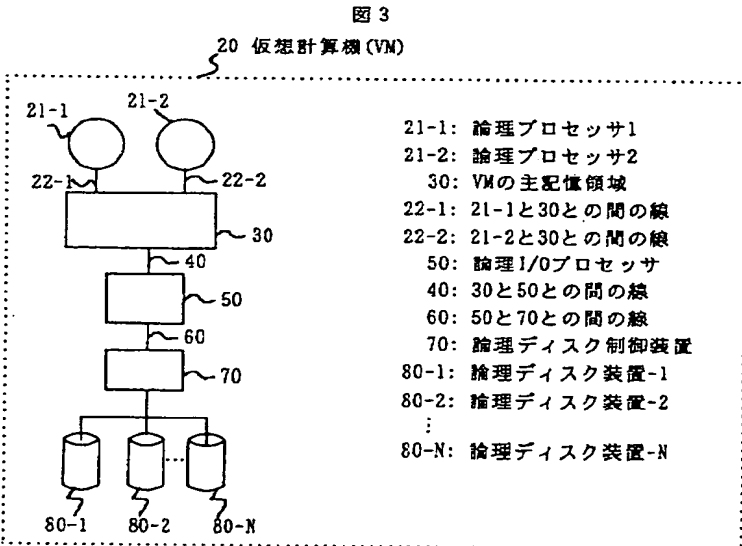
【符号の説明】

400・・・システム初期化処理の部分、401・・・ディスパッチャ、402・・・スケジューラ、403・・・スケジューラ用周期タイマ制御部分、404・・・サービス量(U)計算部分、405・・・キュー操作部分、406・・・制御値(B)設定部分、440・・・走行中の論理プロセッサのリスト、410・・・レディキュー(Ready queue)、420・・・アウトサービスキュー(Out service queue)、430・・・ウェイトキュー(Wait queue)。

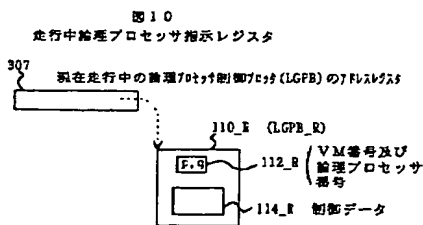
【 図2 】



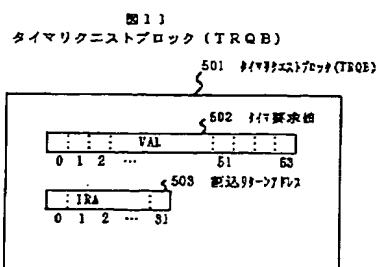
【 図3 】



【 図10 】



【 図11 】



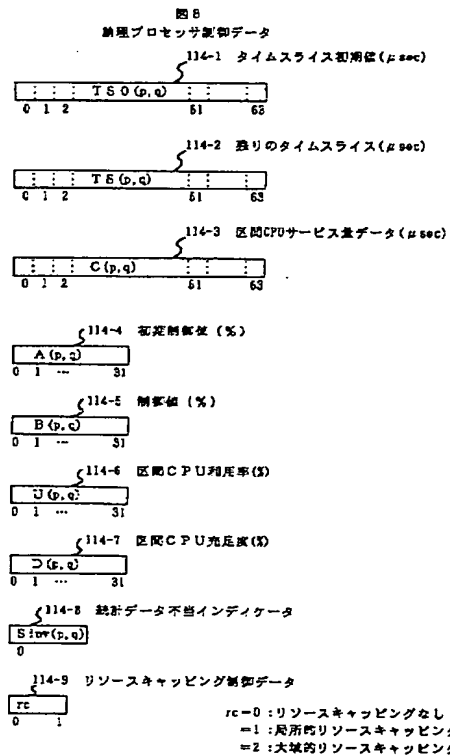
【 図6 】

図6

記号説明

記号	意味
$C(p, q)$	区間における論理プロセッサLGP (p, q) の CPU busy time (μsec)
Δt	区間の時間長さ: どの区間にも同じ大きさとする. (μsec)
A1	VM1のCPU利用率の初期制御値 (%)
A2	VM2のCPU利用率の初期制御値 (%)
\vdots	\vdots
AN	VMNのCPU利用率の初期制御値 (%)
$A(p, q)$	論理プロセッサLGP (p, q) のCPU利用率の初期制御値 (%) $p=1, 2, \dots, q=1, 2, \dots$

【 図8 】



【 図19 】

図19

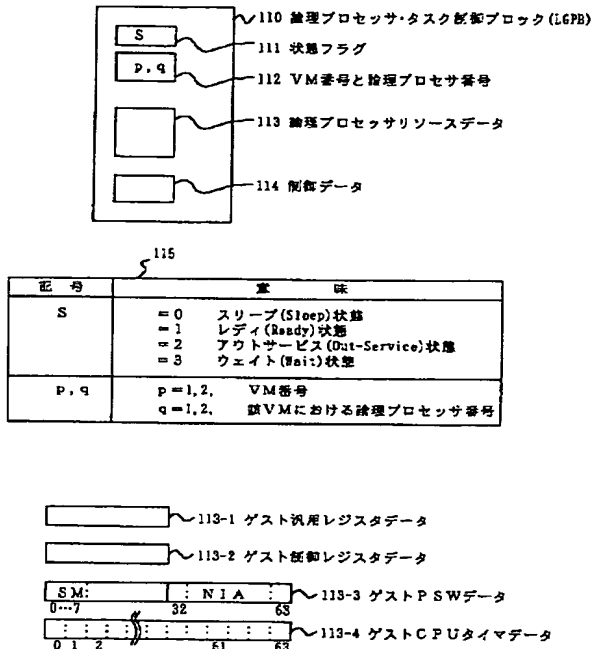
モードレジスタ

305 モードレジスタ

番号	意味
1	= 1 命令プロセッサがVMモード = 0 命令プロセッサが基本モード

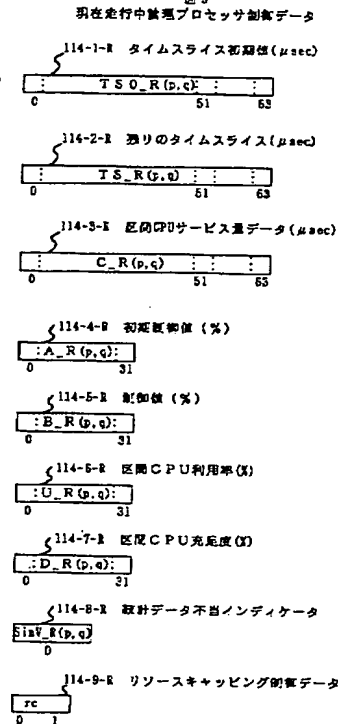
【 図 7 】

図 7



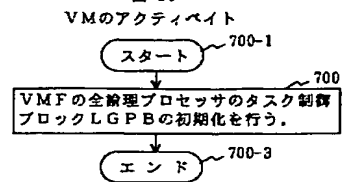
【 図 9 】

図 9

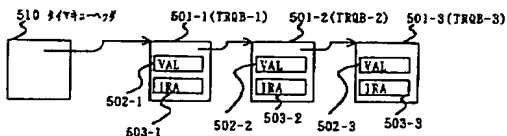


【 図 2 4 】

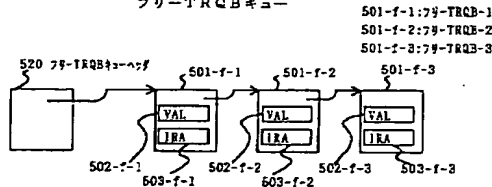
図 24



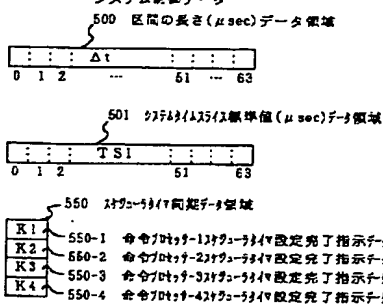
【 図 1 2 】

図 12
タイムリクエストキュー

【 図 1 3 】

図 13
フリー TRQB キュー

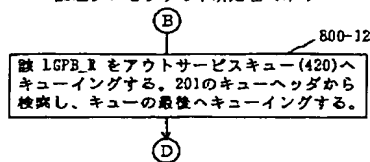
【 図 1 4 】

図 14
システム制御データ

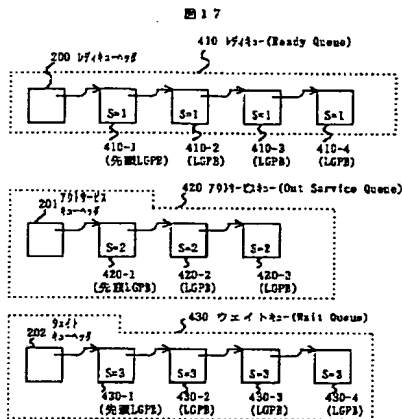
【 図 2 9 】

図 29

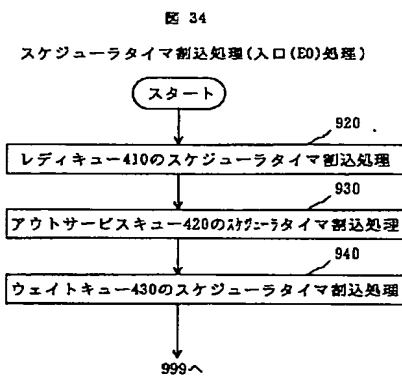
論理プロセッサの中断処理 (3/3)



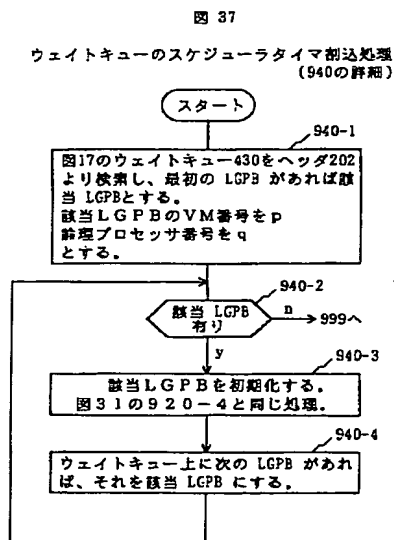
【 図 17 】



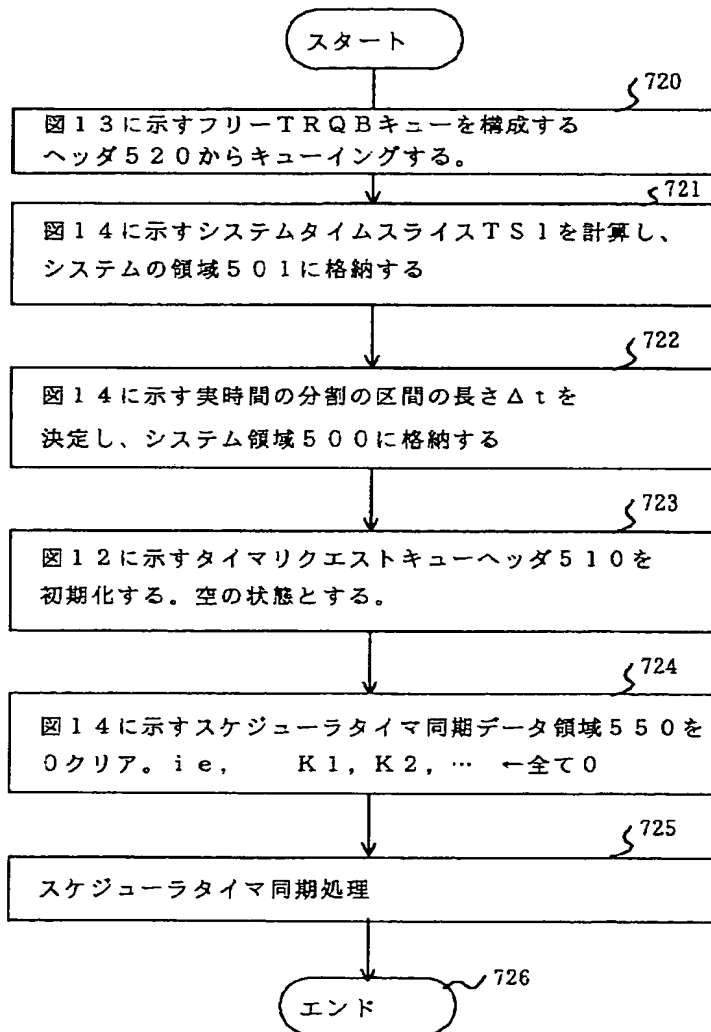
【 図 34 】



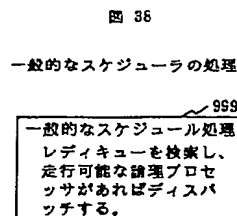
【 図 37 】



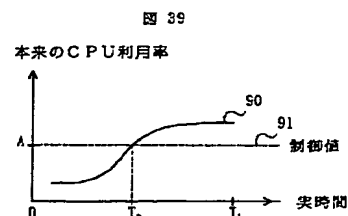
【 図 20 】

図 20
ハイババイザ立ち上げ時の初期化

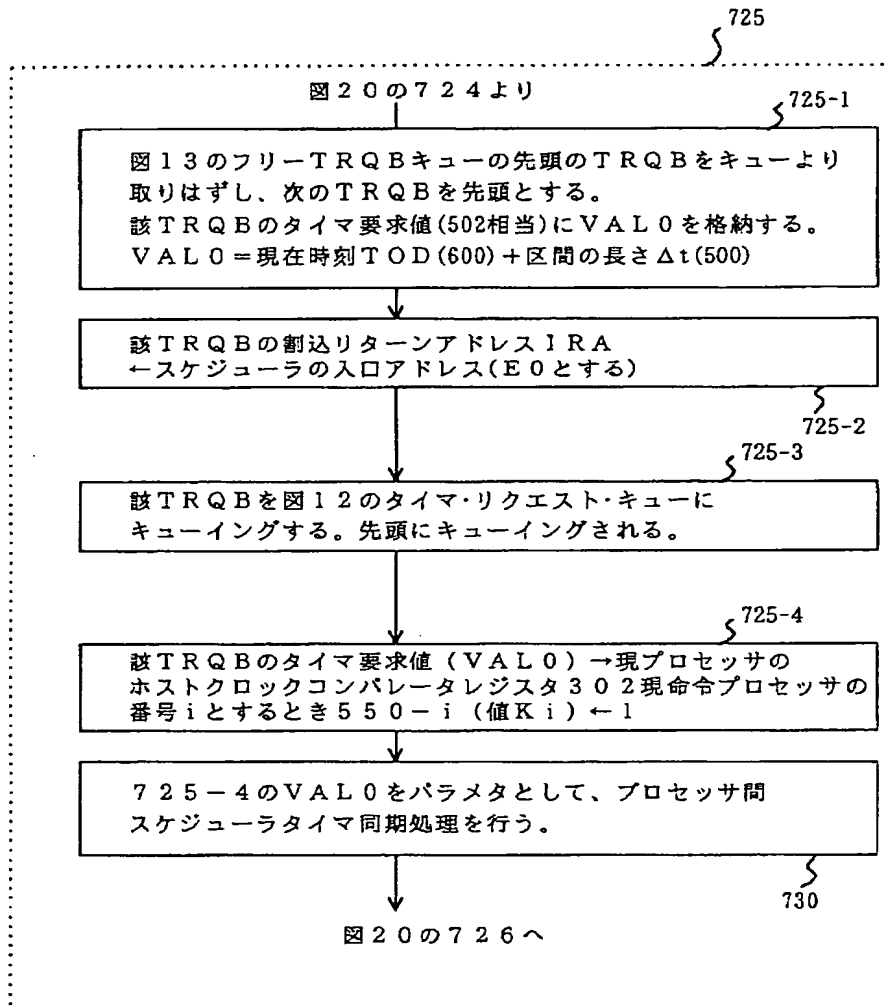
【 図 38 】



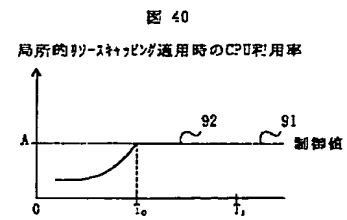
【 図 39 】



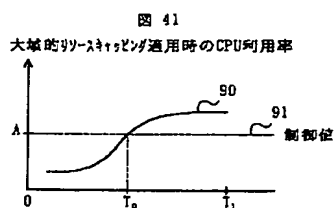
【 図2 1 】

図 2 1
スケジューラタイマ同期処理

【 図4 0 】



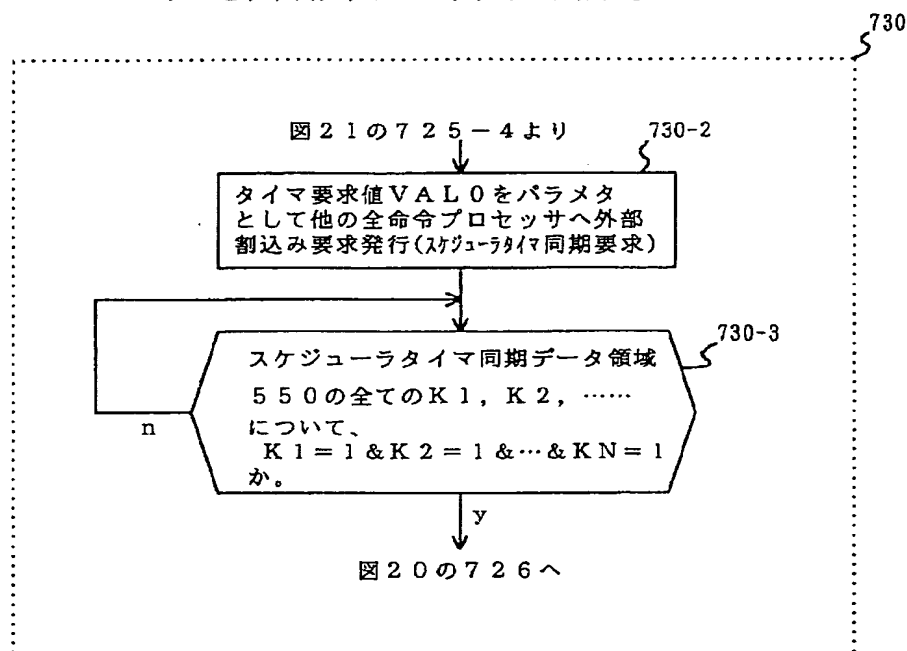
【 図4 1 】



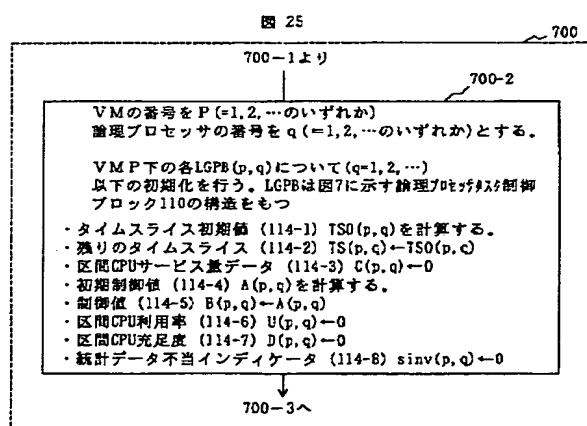
【图22】

22

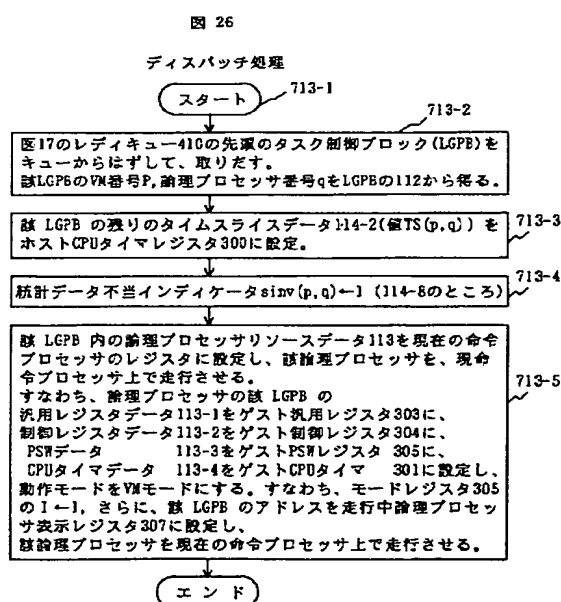
プロセス間スケジューラタイマ同期処理



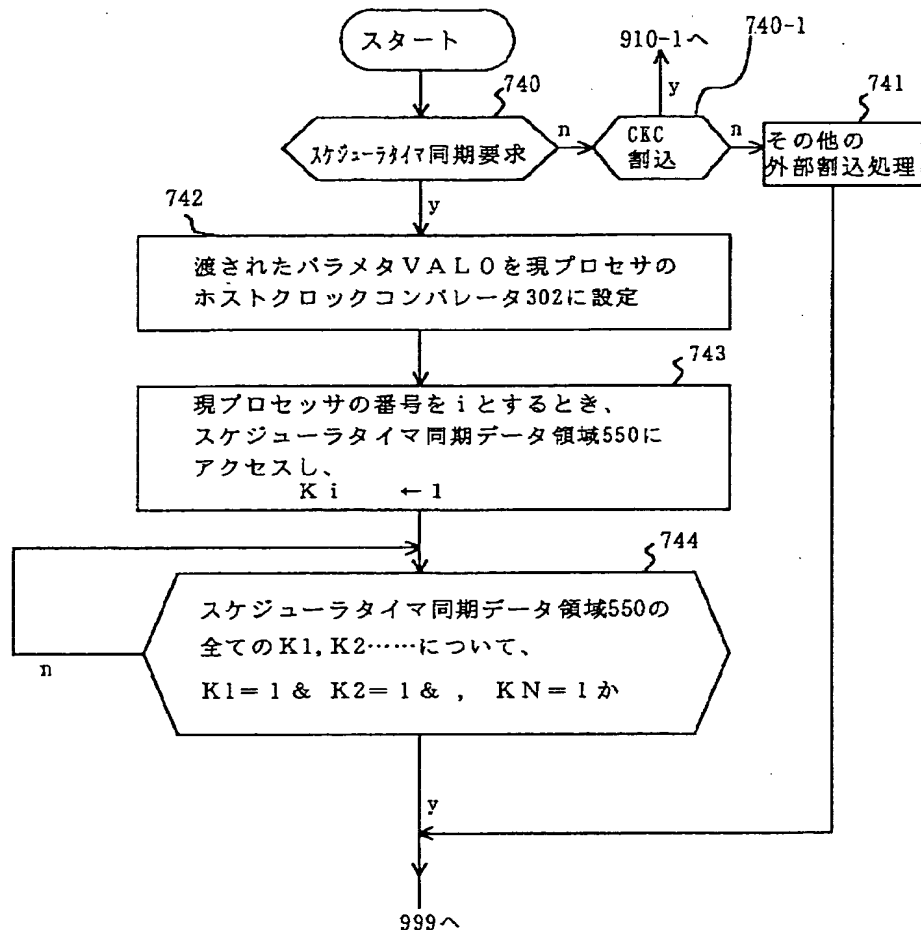
【图25】



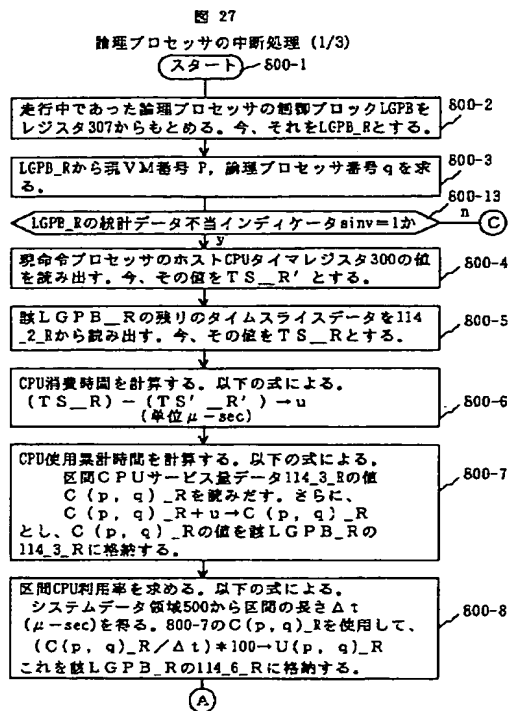
【图26】



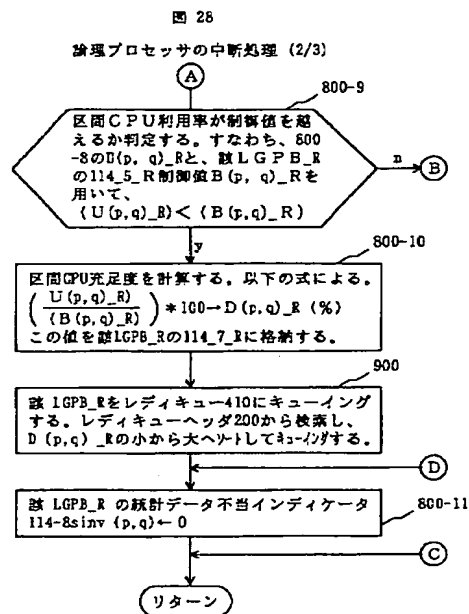
【 図23 】

図 2 3
外部割込処理

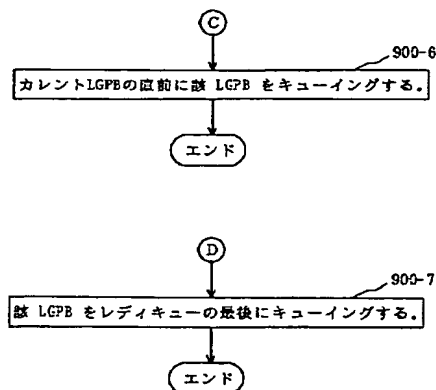
【 図27 】



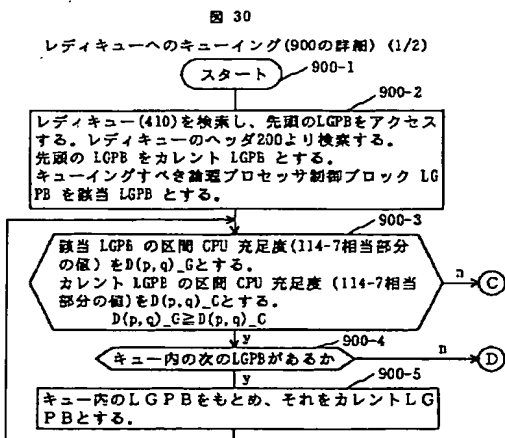
【 図28 】



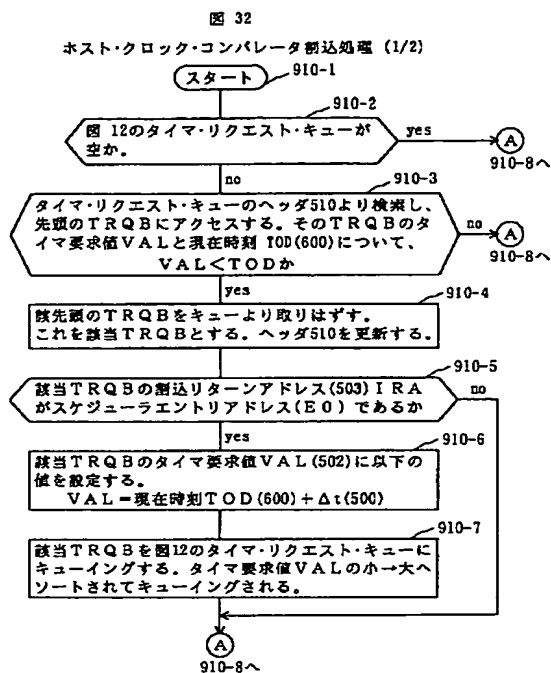
【 図31 】

図 31
レディキューへのキューイング (2/2)

【 図30 】

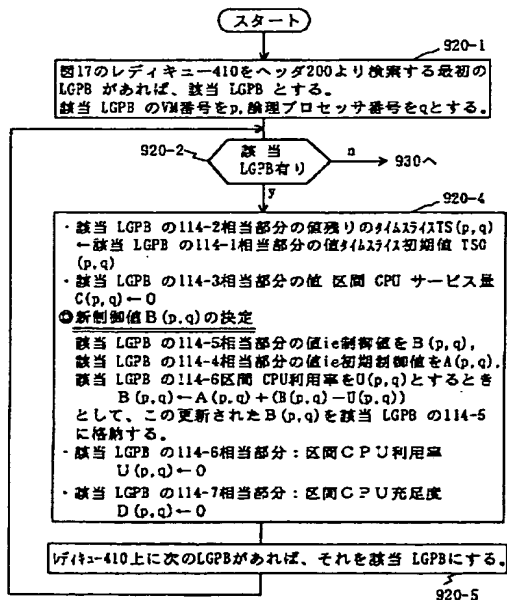


【 図 3 2 】



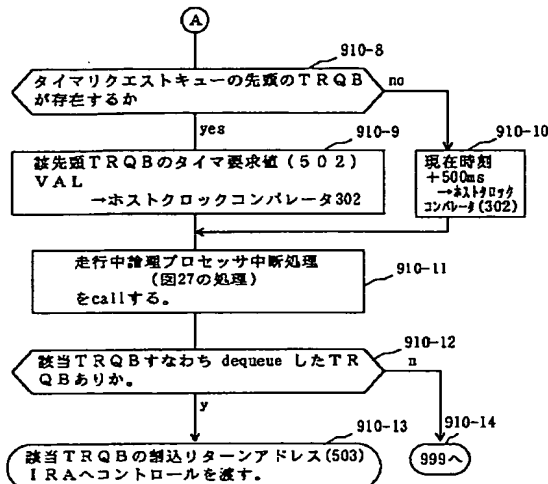
【 図 3 5 】

図 35
レディキューに対するスケジューラタイマ割込処理(920の詳細)



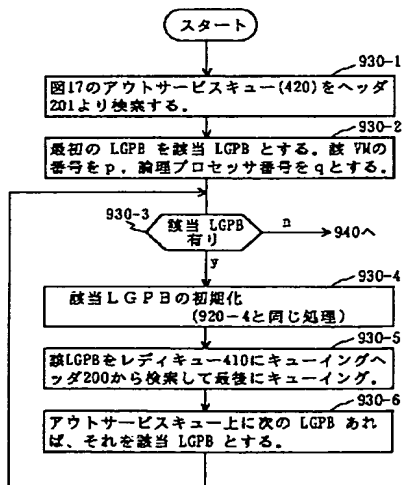
【 図 3 3 】

図 33
ホスト・クロック・コンパレータ割込処理 (2/2)



【 図 3 6 】

図 36
アウトサービスキューのスケジューラタイマ割込処理(930の詳細)



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.